Learning With Incremental Instances and Features

Shilin Gu, Yuhua Qian, *Member, IEEE*, and Chenping Hou, *Member, IEEE*

***Abstract*— In many real-world applications, data may dynamically expand over time in both volume and feature dimensions. Besides, they are often collected in batches (also called blocks). We refer this kind of data whose volume and features increase in blocks as blocky trapezoidal data streams. Current works either assume that the feature space of data streams is fixed or stipulate that the algorithm receives only one instance at a time, and none of them can effectively handle the blocky trapezoidal data streams. In this article, we propose a novel algorithm to learn a classification model from blocky trapezoidal data streams, called learning with incremental instances and features (IIF). We attempt to design highly dynamic model update strategies that can learn from increasing training data with an expanding feature space. Specifically, we first divide the data streams obtained on each round and construct the corresponding classifiers for these different divided parts. Then, to realize the effective interaction of information between each classifier, we utilize a single global loss function to capture their relationship. Finally, we use the idea of ensemble to achieve the final classification model. Furthermore, to make this method more applicable, we directly transform it into the kernel method. Both theoretical analysis and empirical analysis validate the effectiveness of our algorithm.**

***Index Terms*— Blocky trapezoidal data streams, classification, evolvable features, learning with streaming data.**

Nomenclature

$\ell_{t,j}$	Hinge loss of classifier $\mathbf{w}_{t,j}$ on $\mathcal{X}_{t,j}$.
$\tilde{\ell}_{t,j}$	Hinge loss of classifier $\tilde{\mathbf{w}}_j$ on $\mathcal{X}_{t,j}$.
$\tau_{t,j}$	Nonnegative scalar.
$\boldsymbol{\ell}_t$	Vector whose elements are $\ell_{t,j}$.
$\tilde{\boldsymbol{\ell}}_t$	Vector whose elements are $\tilde{\ell}_{t,j}$.
$\boldsymbol{\tau}_t$	Vector whose elements are $\tau_{t,j}$.
$\phi(\cdot)$	Function that maps data from the original input space $\mathbb{R}^M$ to the feature space $\mathbb{F}$.
$\kappa(\cdot,\cdot)$	Kernel function.
ρ	Remoteness of a norm.
B	Budget size.
$\mathcal{X}_t \in \mathbb{R}^{n_t \times d_t}$	Data block obtained on round t.
$\mathcal{X}_{t,j} \in \mathbb{R}^{n_t \times m_j}$	jth part of $\mathcal{X}_t$.
$\mathcal{Y}_t \in \mathbb{R}^{n_t \times 1}$	Label vector of $\mathcal{X}_t$.
$\mathbf{x}_{t,j}^i \in \mathbb{R}^{m_j}$	ith instance of the jth part observed on round t.
$y_{t,j}^i \in \{-1,+1\}$	Label of $\mathbf{x}_{t,j}^i$.
$\hat{y}_{t,j}^i \in \{-1,+1\}$	Predicted label of $\mathbf{x}_{t,j}^i$.
$\mathbf{w}_{t,j} \in \mathbb{R}^{m_j}$	jth linear classifier on round t.
$\tilde{\mathbf{w}}_j \in \mathbb{R}^{m_j}$	Arbitrary vectors with the same dimension as $\mathcal{X}_{t,j}$.
$\hat{\mathbf{x}} \in \mathbb{R}^{d_t}$	Unlabeled instance.
$\hat{\mathbf{x}}_j \in \mathbb{R}^{m_j}$	Vector consisting of elements of $\hat{\mathbf{x}}$ that are in the same feature space of $\mathcal{X}_{t,j}$.

Manuscript received 22 November 2020; revised 7 November 2021, 26 June 2022, and 12 October 2022; accepted 9 January 2023. This work was supported in part by the Key NSF of China under Grant 62136005; and in part by the NSF of China under Grant 61922087, Grant 61906201, and Grant 62006238. *(Corresponding authors: Yuhua Qian; Chenping Hou.)*

Shilin Gu and Chenping Hou are with the College of Science, National University of Defense Technology, Changsha, Hunan 410073, China (e-mail: gslnudt@outlook.com; hcpnudt@hotmail.com).

Yuhua Qian is with the Institute of Big Data Science and Industry, Shanxi University, Taiyuan, Shanxi 030006, China (e-mail: jinchengqyh@126.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3236479.

Digital Object Identifier 10.1109/TNNLS.2023.3236479

I. Introduction

In many real-world applications, we have to deal with the situation where both data volume and feature dimension dynamically keep expanding, such as the text classification with an increasing number of documents and text vocabulary or the classification of data collected by continuously installed detectors in the environment monitoring. Also, we usually receive a batch of data instead of one instance at the same time since the detectors collect data for a certain period of time before being retrieved. This kind of data is referred to as blocky trapezoidal data streams. As shown in Fig. 1, for ecological environment monitoring, we can install many detectors to collect data, where each detector corresponds to a batch of features. With the development of technology, we can deploy more advanced detectors to collect more features of the environment, that is, both the data collected by detectors and the features of data increase over time.

The setting of online learning is relatively close to blocky trapezoidal data stream learning. Traditional online learning algorithms mainly solve the problem where either the training instances arrive sequentially, but the feature space is fixed [1], [2], [3], [4], [5], [6], or the features arrive continuously as streams, but the training set is fixed [7], [8], [9], [10], [11]. Obviously, the traditional online learning algorithms, such as the perceptron algorithm [6], the passive–aggressive (PA) algorithm [4], and the online streaming feature selection (OSFS) algorithm [9], cannot be directly applied to handle the blocky trapezoidal data streams because their model update criteria are designed either in the case of increasing data volume or increasing feature dimension, without considering the increase in both aspects at the same time. To deal with the blocky trapezoidal data streams using traditional online learning algorithms, a straightforward approach is to take advantage of the newly obtained data and learn a new model for classification. However, this approach may have two deficiencies. First, the newly coming data are usually scarce, which might be insufficient to build a superior predictive model. Second, the

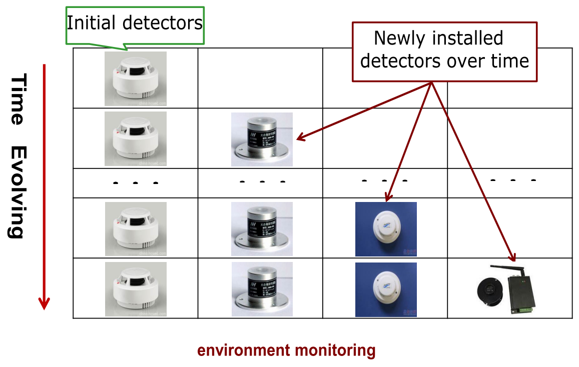

Fig. 1. Illustration of blocky trapezoidal data stream. In the first row, we only have a few initial detectors, and as time evolves, more advanced detectors are installed. Each column represents the features collected by a batch of detectors that are installed at the same time. Each row represents the data composed of features that are collected by all current detectors. Therefore, the volume and dimensions of blocky trapezoidal data streams are increasing over time.

newly built model ignores the previously collected data, which cannot fully exploit the information from the historical data and causes big waste.

Therefore, the main challenge in dealing with blocky trapezoidal data streams is how to design highly dynamic model update strategies that can learn from increasing training data with an expanding feature space. Recently, a few studies have explored solving trapezoidal data streams [12], [13], [14]. They adopt different feature division or projection strategies to construct dynamic models, which make good improvements in dealing with trapezoidal data streams. However, all these studies are not designed for the situation where data arrive in blocks and have no corresponding theoretical guarantee.

In this article, we propose a new learning with incremental instances and features (IIF) algorithm to solve blocky trapezoidal data streams, which has a broader application scenario than trapezoidal data stream learning. We design a novel strategy to learn a highly dynamic classification model from blocky trapezoidal data streams and then use the classifier for prediction. In our setting, each group of newly installed detectors forms a set, and the algorithm receives a batch of data collected by current multiple detector sets at each iteration. For the features collected by each set of detectors, we treat them as a batch of independent data, and then, we construct a classification model on such data and update it by following the update rule used in online multitask learning [15]; the relationship between these classifiers can be captured by a single global loss function. Specifically, each classifier can make a prediction on each independent data, and the predictive quality of each classifier is associated with its loss; then, we can combine these loss values by using a global loss function. Finally, we use the idea of ensemble to achieve the final classification model. The proposed algorithm uses an additive update rule, so we directly transform it into a kernel method for description in this article. Theoretical and empirical studies validate the performance of the proposed algorithm.

It is worthwhile to summarize the main contributions of the proposed approach as follows.

1) We propose a new method IIF to deal with blocky trapezoidal data streams where data may expand in both volume and feature dimensions. To the best of our knowledge, there is a very little study in the literature that is specially designed for this kind of data stream.
2) We design a novel strategy to learn a highly dynamic classification model from blocky trapezoidal data streams. We theoretically analyze the performance bounds of the proposed algorithm.
3) Extensive experiments on both synthetic and real-world datasets demonstrate the effectiveness of IIF.

The remainder of this article is organized as follows. Section II introduces the related work. Section II introduces the setting of the problem. The proposed algorithm is presented in Section IV. Section V provides the corresponding analysis and the detailed proofs of our theorems. Section VI reports the experimental results. We conclude the work in Section VII.

II. Related Work

In this article, we aim to learn a classification model from blocky trapezoidal data streams, which is closely related to online learning from fixed feature space and online learning from dynamic feature space.

A. Learning From Fixed Feature Space

This pattern refers to traditional online learning algorithms, where training instances arrive in sequence with feature space fixed, i.e., every instance has the same number of features [16]. There are many studies on traditional online learning [1], [2], [17]. Generally, we can divide them into two categories, linear and nonlinear online learning algorithms.

The linear online learning algorithms achieve satisfactory performance when the training instances are linearly separable [1], [4], [6]. The perceptron algorithm [6] and PA algorithm [4] are two well-known linear algorithms. At each round, the perceptron updates the model only when the model makes a wrong prediction. However, the PA algorithm aggressively updates the model whenever the loss is nonzero. Therefore, PA algorithms usually have a better performance than perceptron. Recently, people have proposed many new linear online learning algorithms [17], [18], [19], [20], and they all improve online learning in different aspects.

When data are linearly inseparable, we often need to map the data to another high-dimensional reproducing kernel Hilbert space, and then, the newly mapped data are linearly separable in the new space [21]. These algorithms are referred to as kernel-based online learning. Different online kernel methods, such as the kernelized perceptron [22] and kernelized OGD [23], have been proposed. Although these methods perform better than linear models, they have to face the "curse of kernelization" [24] in large-scale learning problems, that is, the growing unbounded number of support vectors (SVs). Recently, many budget online learning algorithms have been proposed to address this challenge [25], [26], [27], [28], [29], [30], [31]. For example, Wang and Vucetic [26] proposed a budgeted PA algorithm by introducing an additional constraint to the original PA optimization problem. Furthermore, unlike the regular budget online kernel learning, Lu et al. [27] proposed two efficient methods Fourier online gradient descent (FOGD) and Nystrom online gradient descent (NOGD) to explore a new approach to kernel functional approximation techniques. Since these traditional online learning algorithms all assume that the feature space they learn from remains the same, they cannot handle the blocky trapezoidal data streams.

B. Learning From Dynamic Feature Space

The dynamic feature space contains incremental feature space and varying feature space. The most relevant to our work is trapezoidal data stream learning, which belongs to data stream with incremental feature space. Zhang et al. [12] were the first to deal with trapezoidal data stream. They proposed OL$_{\text{SF}}$ with its two variants OL$_{\text{SF}}$-I and OL$_{\text{SF}}$-II. Specifically, Zhang [12] first divided the features of the current training instance into historical features and new features. Then, a classifier updates historical features and new features by following different update rules used in online learning. Wu et al. [9] proposed OSFS algorithms to solve online streaming feature selection, where features arrive as streams, but the training set is fixed [32]. Recently, a few works have been proposed to learn from data streams with varying feature space, where features would vanish or occur over time [33], [34], [35], [36]. Hou et al. [34] proposed the feature evolvable streaming learning (FESL) algorithm; it first recovers historical features by a mapping function learned in the period where both historical features and new features exist, and then, it learns two models from features of the above two parts. Finally, ensemble learning is used to make the final prediction. Based on FESL, Zhang et al. [35] and Hou [36] conducted in-depth exploration and expansion of the FESL scenario and proposed evolving discrepancy minimization (EDM) [35] and prediction with unpredictable feature evolution (PUFE) [36] algorithms to handle different situations in data streams with varying feature space. Beyazit et al. [13] proposed online learning from varying features (OLVF) to project the existing model and the current instance onto shared feature space and make a prediction. He et al. [37] proposed the generative learning with streaming capricious data (GLSC) algorithm, which establishes the relationships between historical and new features by constructing a model on a universal feature space. Nevertheless, all these above studies are not designed for the situation where data arrive in blocks.

III. Preliminaries

In this article, we focus on the binary classification problem. The multiclass cases could be easily solved by using one-versus-one [38] or one-versus-rest [39] strategies.

For blocky trapezoidal data stream learning, data blocks are obtained in a sequence of rounds. On round t, we obtain data block $(\mathcal{X}_t, \mathcal{Y}_t)$ with n_t instances, where $\mathcal{X}_t \in \mathbb{R}^{n_t \times d_t}$ and $\mathcal{Y}_t \in \mathbb{R}^{n_t}$. We divide these instances into t parts $\{(\mathcal{X}_{t,1}, \mathcal{Y}_t), \ldots, (\mathcal{X}_{t,t}, \mathcal{Y}_t)\}$ according to the way in Fig. 2, where $\mathcal{X}_{t,j} \in \mathbb{R}^{n_t \times m_j}$, $j = 1, \ldots, t$, and $d_t = m_1 + \cdots + m_t$. Let $\mathbf{w}_{t,j} \in \mathbb{R}^{m_j}$ denote the linear classifier for the jth data part on round t. The algorithm maintains t separate classifiers in its internal memory on round t and updates them from round to round. Specifically, each divided part $(\mathcal{X}_{t,j}, \mathcal{Y}_t)$ contains n_t instances $(\mathbf{x}_{t,j}^i, y_t^i)$, where $i = 1, \ldots, n_t$, $\mathcal{Y}_t = (y_t^1, \ldots, y_t^{n_t})$. Note that y_t^i and $y_{t,j}^i$ are the real labels of the ith instances in $\mathcal{X}_t$ and $\mathcal{X}_{t,j}$, respectively. Therefore, we have $y_t^i = y_{t,j}^i$. The

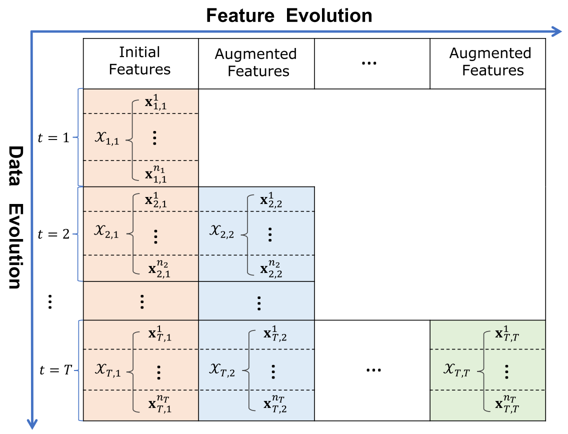

Fig. 2. Proposed algorithm notations. The features and instances dynamically expand, and newly augmented features in different periods are marked by different colors (red, blue, green, and so on). We divide the data streams according to the color of the feature and learn the corresponding classification models for the divided parts.

algorithm first uses the classifier $\mathbf{w}_{t,j}$ to predict the binary labels $\hat{y}_{t,j}^1, \ldots, \hat{y}_{t,j}^{n_t}$, where $\hat{y}_{t,j}^i = \text{sign}(\mathbf{w}_{t,j}^T \cdot \mathbf{x}_{t,j}^i)$. Then, the correct labels of the respective $y_t^1, \ldots, y_t^{n_t}$ are revealed and the classifier $\mathbf{w}_{t,j}$ can be updated based on the prediction loss.

Generally, for an online binary classification task, the update rule of $\mathbf{w}$ in perceptron is given as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \mathbf{x}_t$$

where $\alpha_t = y_t$ is the weight. This linear model can only make accurate prediction when the instances are linearly separable. However, the data may contain complicated pattern and may be linearly inseparable in practice. This problem can be solved by using a mapping function $\phi(\cdot)$ to map $\mathbf{x}$ from the original input space $\mathbb{R}^M$ to the feature space $\mathbb{F}$. Then, the new training instances $\phi(\mathbf{x})$ are linearly separable in the new feature space $\mathbb{F}$. We define SV as the instance whose coefficient α is nonzero. Obviously, $\mathbf{w}$ is the weighted sum of SVs, i.e., $\mathbf{w} = \sum_{i \in \mathcal{SV}} \alpha_i \phi(\mathbf{x}_i)$, where $\mathcal{SV}$ is the set of SV's and i is its index. Thus, the predictor $f(\mathbf{x})$ is

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) = \sum_{i \in \mathcal{SV}} \alpha_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}). \tag{1}$$

Instead of directly computing $\phi(\mathbf{x})$, a kernel function $\kappa(\cdot, \cdot)$ is introduced such that $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$, and then, the predictor $f(\mathbf{x})$ can be denoted as

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) = \sum_{i \in \mathcal{SV}} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}). \tag{2}$$

In summary, the notations used in this article are listed in the Nomenclature and we will explain the concrete meaning when it is first used.

IV. Our Proposed Approach

In this section, we formally present the proposed IIF algorithm. IIF mainly contains two steps: one is the update step and the other one is the budget step.

A. Update Strategy

As stated in Section III, on round t, the algorithm divides the observed data block $(\mathcal{X}_t, \mathcal{Y}_t)$ into t parts $\{(\mathcal{X}_{t,1}, \mathcal{Y}_t), \ldots, (\mathcal{X}_{t,t}, \mathcal{Y}_t)\}$ according to the way in Fig. 2, where $\mathcal{X}_{t,j} \in \mathbb{R}^{n_t \times m_j}$, $j = 1, \ldots, t$, and $d_t = m_1 + \cdots + m_t$. Classifiers for each of the parts $\mathcal{X}_{t,j}$ are maintained in the algorithm's internal memory and updated from round to round. For each part $\mathcal{X}_{t,j} \in \mathbb{R}^{n_t \times m_j}$, the corresponding linear classifier is $\mathbf{w}_{t,j} \in \mathbb{R}^{m_j}$. We use $\mathbf{w}_{t,j}$ to predict the binary labels $\hat{y}_{t,j}^1, \ldots, \hat{y}_{t,j}^{n_t}$ of dataset $\mathcal{X}_{t,j}$, where $\hat{y}_{t,j}^i = \text{sign}(\mathbf{w}_{t,j}^T \cdot \phi(\mathbf{x}_{t,j}^i))$. Then, the correct labels of the respective $y_t^1, \ldots, y_t^{n_t}$ are revealed and the classifier $\mathbf{w}_{t,j}$ can be updated based on the prediction loss. Usually, to penalize incorrect predictions, we can use a hinge loss as the loss function, which makes the learner update the prediction model from $\mathbf{w}_{t,j}$ to $\mathbf{w}_{t+1,j}$ by some strategy whenever the loss is nonzero.

In this article, we slightly modify the loss function and use an average hinge-loss function to penalize incorrect predictions for part $\mathcal{X}_{t,j}$. The loss associated with the j'th part on round t is defined to be

$$\ell_{t,j} = \left[\frac{1}{n_t}\sum_{i=1}^{n_t}\left(1 - y_{t,j}^i \cdot \mathbf{w}_{t,j}^T \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right)\right]_+ . \tag{3}$$

where $[a]_+ = \max\{0, a\}$, $j = 1, 2, \ldots, t$. In our setting, we assume that the algorithm receives a batch of data at each iteration; therefore, it is inefficient to calculate the prediction loss and update the classifier for each instance. Conversely, performing prediction loss calculation and classifier update on a batch of data can reduce the frequency of classifier update and reduce the running time to a certain extent, which has more practical application significance. Besides, at the beginning of online learning, the learned classification model can be poor, and using average loss can reduce the update bias caused by misclassification to a certain extent.

In this article, we use an additive update rule as follows to update the classifiers:

$$\mathbf{w}_{t+1,j} = \mathbf{w}_{t,j} + \frac{1}{n_t}\tau_{t,j}\sum_{i=1}^{n_t} y_{t,j}^i \cdot \phi\left(\mathbf{x}_{t,j}^i\right) \tag{4}$$

where $\tau_{t,j}$ is a nonnegative scalar, and the determination of its value will be discussed later. By adopting the above update strategy, we can construct a separate classifier for each of the parts $\mathcal{X}_{t,j}$ on round t. Then, we are able to achieve the fused classifier $\mathbf{w}_{t+1}$ by using the ensemble idea at the end of each round

$$\mathbf{w}_{t+1} = \left[\mathbf{w}_{t+1,1}^T, \ldots, \mathbf{w}_{t+1,t}^T\right]^T \tag{5}$$

where $\mathbf{w}_{t+1}$ fully absorbs the information of the previous t data blocks so that it can take advantage of more features in building a superior predictive model.

However, when we need to predict an unlabeled instance $\hat{\mathbf{x}} \in \mathbb{R}^{m_1+\cdots+m_t}$ on round t, it is usually hard to directly compute $\phi(\hat{\mathbf{x}})$, and we can directly update $f_{t+1,j}$ instead of $\mathbf{w}_{t+1,j}$. We first have the following definition:

$$\hat{\mathbf{x}}_j = \prod_{\mathcal{X}_{t,j}} \hat{\mathbf{x}} \in \mathbb{R}^{m_j} \tag{6}$$

where $\hat{\mathbf{x}}_j \in \mathbb{R}^{m_j}$ represents a projection of the feature space from dimension d_t to dimension m_j, and it is a vector consisting of elements of $\hat{\mathbf{x}}$ that are in the same feature space of $\mathcal{X}_{t,j}$. From (6), we can easily know that

$$\hat{\mathbf{x}} = \left[\hat{\mathbf{x}}_1^T, \ldots, \hat{\mathbf{x}}_t^T\right]^T. \tag{7}$$

Then, we have

$$\begin{aligned} f_{t+1,j}\left(\hat{\mathbf{x}}_j\right) &= \mathbf{w}_{t+1,j}^T \cdot \phi\left(\hat{\mathbf{x}}_j\right) \\ &= \left(\mathbf{w}_{t,j} + \frac{1}{n_t}\tau_{t,j}\sum_{i=1}^{n_t} y_t^i \phi\left(\mathbf{x}_{t,j}^i\right)\right)^T \phi\left(\hat{\mathbf{x}}_j\right) \\ &= \mathbf{w}_{t,j}^T \cdot \phi\left(\hat{\mathbf{x}}_j\right) + \frac{1}{n_t}\tau_{t,j}\sum_{i=1}^{n_t} y_t^i \kappa\left(\mathbf{x}_{t,j}^i, \hat{\mathbf{x}}_j\right) \\ &= f_{t,j}\left(\hat{\mathbf{x}}_j\right) + \frac{1}{n_t}\tau_{t,j}\sum_{i=1}^{n_t} y_t^i \kappa\left(\mathbf{x}_{t,j}^i, \hat{\mathbf{x}}_j\right). \end{aligned} \tag{8}$$

After we get each classification model $f_{t+1,j}$, we can achieve the fused classifier f_{t+1} on round t as follows to predict the unlabeled instance $\hat{\mathbf{x}}$:

$$f_{t+1}\left(\hat{\mathbf{x}}\right) = \sum_{j=1}^{t} \mathbf{w}_{t+1,j}^T \cdot \phi\left(\hat{x}_j\right) = \sum_{j=1}^{t} f_{t+1,j}\left(\hat{x}_j\right). \tag{9}$$

Note that our method does not require accessing all the past training data during both the training and testing phases. In fact, when we use the linear kernel in our method, there is no need for data storage. Concretely, as can be seen from (4), (8), and (9), the mapping function $\phi(\cdot)$ plays an important role in computing $\mathbf{w}$ and f $(f = \mathbf{w}^T \cdot \phi(\mathbf{x}))$. From (8), we can know that

$$f_{t+1,j}\left(\hat{\mathbf{x}}_j\right) = \mathbf{w}_{t,j}^T \cdot \phi\left(\hat{\mathbf{x}}_j\right) + \frac{1}{n_t}\tau_{t,j}\sum_{i=1}^{n_t} y_t^i \kappa\left(\mathbf{x}_{t,j}^i, \hat{\mathbf{x}}_j\right). \tag{10}$$

With this observation, on the one hand, if we use the linear kernel, whose mapping function $\phi(\cdot)$ has one explicit expression, i.e., $\phi(\mathbf{x}_i) = \mathbf{x}_i$ [40] and $\kappa(\mathbf{x}_i, \mathbf{x}) = \mathbf{x}_i^T \cdot \mathbf{x}$, we will have

$$\begin{aligned} f_{t+1,j}\left(\hat{\mathbf{x}}_j\right) &= \mathbf{w}_{t+1,j}^T \cdot \phi\left(\hat{\mathbf{x}}_j\right) \\ &= \mathbf{w}_{t,j}^T \cdot \hat{\mathbf{x}}_j + \frac{1}{n_t}\tau_{t,j}\sum_{i=1}^{n_t} y_t^i \left(\mathbf{x}_{t,j}^i\right)^T \cdot \hat{\mathbf{x}}_j \\ &= \left(\mathbf{w}_{t,j} + \frac{1}{n_t}\tau_{t,j}\sum_{i=1}^{n_t} y_t^i \mathbf{x}_{t,j}^i\right)^T \cdot \hat{\mathbf{x}}_j. \end{aligned} \tag{11}$$

In this case, $\mathbf{w}_{t+1,j} = \mathbf{w}_{t,j} + (1/n_t)\tau_{t,j}\sum_{i=1}^{n_t} y_{t,j}^i \mathbf{x}_{t,j}^i$ and we can compute $f_{t+1,j}(\hat{\mathbf{x}}_j) = \mathbf{w}_{t+1,j}^T \cdot \hat{\mathbf{x}}_j$ directly, with no need to compute the inner products $\kappa(\mathbf{x}_{t,j}^i, \hat{\mathbf{x}}_j)$. In other words, the information of past data has been stored in the vector $\mathbf{w}_{t,j}$. We only need to maintain this vector, with no need to store past data in the whole learning process. On the other hand, if the mapping function $\phi(\cdot)$ does not have an explicit expression, such as that in the Gaussian kernel, we cannot update $\mathbf{w}_{t+1,j}$ directly. In this case, we only need to store part of the past data whose coefficient $\tau_{t,j} \neq 0$, called SVs. The nonsupport vectors $\mathcal{X}_{t,j}$ with $\ell_{t,j} = 0$ make $\tau_{t,j} = 0$ as $\tau_{t,j} = C\ell_{t,j}/\|\boldsymbol{\ell}_t\|_2$ in (19) and we need not to store them. In the following, we propose a budget strategy to further bound the number of SVs.

Next, we give the derivation process of $\tau_{t,j}$. As can be seen from (3), we can get t losses on round t. Generally, in machine learning, we need to fully extract the intrinsic information between the features and effectively merge the extracted information to learn a better classifier. Usually, the exchange of information between features can be achieved through losses. For the blocky trapezoidal data streams, different parts of the data are conducive to the final classification goal. Therefore, the main idea of the proposed algorithm is to effectively merge these losses $\ell_{t,j}$ obtained above to update the t classifiers on round t and suffer the smallest possible cumulative loss throughout T rounds.

Specifically, since the prediction of each part $\mathcal{X}_{t,j}$ is associated with its own individual loss, we can capture the relationship between the different parts $\mathcal{X}_{t,j}$ after getting t losses on round t by using a single global loss function, which is able to evaluate the predictive quality of individual classifier for each data part. Then, we can merge the information of different parts of the obtained data streams by the above strategy and effectively update each independent classifier. The details of the update strategy are given as follows.

We denote $\boldsymbol{\ell}_t = (\ell_{t,1}, \ldots, \ell_{t,t})$ the loss vector, where $\ell_{t,j}$ is the individual loss values of the data part $\mathcal{X}_{t,j}$, $j = 1, \ldots, t$. Then, we are able to define the global loss function as follows:

$$\mathcal{L}(\boldsymbol{\ell}_t) = \|\boldsymbol{\ell}_t\|_2 = \left(\sum_{j=1}^{t} \left|\ell_{t,j}\right|^2\right)^{1/2}. \tag{12}$$

Now, we can update each classifier $\mathbf{w}_{t,j}$ by taking advantage of $\boldsymbol{\ell}_t$. As shown in (4), the key problem is how to use $\boldsymbol{\ell}_t$ to achieve the step size $\tau_{t,j}$. Generally, the online learning algorithms need to balance a tradeoff between retaining the information acquired on previous rounds and modifying the hypotheses based on the new instances obtained on the current round. Nevertheless, there are multiple classifiers that need to be updated at every round in our setting, and if we balance this tradeoff individually for each of the classifier updates, then the information exchange between different parts $\mathcal{X}_{t,j}$ will become more difficult. Therefore, we jointly balance this tradeoff for all of the parts. In this way, we are able to make more aggressive modifications to some of the hypotheses at the expense of others.

In order to realize the above idea and create a relationship between $\boldsymbol{\ell}_t$ and $\tau_{t,j}$, we set $\tau_{t,j}$ to be

$$\boldsymbol{\tau}_t = \operatorname*{arg\,max}_{\boldsymbol{\tau}:\|\boldsymbol{\tau}\|^* \leq C} \boldsymbol{\tau} \cdot \boldsymbol{\ell}_t \tag{13}$$

where $\|\cdot\|^*$ is the dual norm of norm $\|\cdot\|$ and $C > 0$ is a constant. The dual norm of $\|\boldsymbol{\ell}_t\|$ is defined as

$$\|\boldsymbol{\ell}_t\|^* = \max_{\mathbf{v}\in\mathbb{R}^t} \frac{\boldsymbol{\ell}_t \cdot \mathbf{v}}{\|\mathbf{v}\|} = \max_{\mathbf{v}\in\mathbb{R}^t:\|\mathbf{v}\|=1} \boldsymbol{\ell}_t \cdot \mathbf{v}. \tag{14}$$

From (14), we can easily derive that for any $\boldsymbol{\ell}_t$, $\mathbf{v} \in \mathbb{R}^t$, it holds that

$$\boldsymbol{\ell}_t \cdot \mathbf{v} \leq \|\boldsymbol{\ell}_t\|^* \|\mathbf{v}\|. \tag{15}$$

In this article, $\|\cdot\|$ is a p-norm and we set $p = 2$, so (15) is also called the Cauchy–Schwartz inequality. We need to know that the dual of $\|\cdot\|^*$ is the original norm $\|\cdot\|$ and that the dual of an absolute norm is also an absolute norm [41]. In order to specify the exact value of $\tau_{t,j}$ we suppose that for $\forall\ 1 \leq t \leq T, 1 \leq j \leq t$, we have $\tau_{t,j} \geq 0$, $\|\boldsymbol{\tau}_t\|^* \leq C$, and $\ell_{t,j} = 0 \Rightarrow \tau_{t,j} = 0$. The above three properties of $\tau_{t,j}$ are called nonnegativity, boundedness, and conservativeness.

Here, we introduce another definition, called the remoteness of a norm $\|\cdot\|$, which plays an important role in the subsequent theoretical analysis. It is defined as

$$\rho(\|\cdot\|, k) = \max_{\mathbf{u}\in\mathbb{R}^k} \frac{\|\mathbf{u}\|_2}{\|\mathbf{u}\|}.$$

Geometrically, $\rho(\|\cdot\|, k)$ is the Euclidean length of the longest vector, which is contained in the unit ball of $\|\cdot\|$. For example, for any p-norm $\|\cdot\|_p$ with $p \geq 2$, $\rho(\|\cdot\|_p, k) = k^{1/2-1/p}$. In the subsequent analysis, we abbreviate $\rho(\|\cdot\|^*, k)$ by ρ for the remoteness of the dual norm $\|\cdot\|^*$ when $\|\cdot\|^*$ and k are obvious from the context.

According to (15) and the definition of remoteness, we have that $\boldsymbol{\tau}_t \cdot \mathbf{u} \leq \|\boldsymbol{\tau}_t\|^*\|\mathbf{u}\|$, and then, we have $\|\boldsymbol{\tau}_t\|_p^2 \cdot \|\mathbf{u}\|_p^2 \leq (\|\boldsymbol{\tau}_t\|^*)^2\|\mathbf{u}\|_p^2$. Since $\|\mathbf{u}\|_p^2$ is an absolute norm, we divide both sides of the inequality by $\|\mathbf{u}\|_p^2$ and have that

$$\begin{aligned}\|\boldsymbol{\tau}_t\|_p^2 &\leq \left(\|\boldsymbol{\tau}_t\|^*\right)^2 \frac{\|\mathbf{u}\|_p^2}{\|\mathbf{u}\|_p^2} \leq \left(\|\boldsymbol{\tau}_t\|^*\right)^2 \frac{\|\mathbf{u}\|_2^2}{\|\mathbf{u}\|_p^2} \\ &\leq \left(\|\boldsymbol{\tau}_t\|^*\right)^2 \max_{\mathbf{u}\in\mathbb{R}^t} \frac{\|\mathbf{u}\|_2^2}{\|\mathbf{u}\|_p^2} = \left(\|\boldsymbol{\tau}_t\|^*\right)^2 \rho^2.\end{aligned} \tag{16}$$

Using (14), we obtain the dual norm of $\|\cdot\|^*$

$$\|\boldsymbol{\ell}_t\|^{**} = \max_{\boldsymbol{\tau}:\|\boldsymbol{\tau}\|^* \leq 1} \boldsymbol{\tau} \cdot \boldsymbol{\ell}_t. \tag{17}$$

Since $\|\cdot\|$ and $\|\cdot\|^{**}$ are equivalent and $\|\boldsymbol{\tau}/C\|^* = \|\boldsymbol{\tau}\|^*/C$, we can easily derive from (13) that $\boldsymbol{\tau}_t$ satisfies

$$\boldsymbol{\tau}_t \cdot \boldsymbol{\ell}_t = C\|\boldsymbol{\ell}_t\|. \tag{18}$$

Then, we can easily know that

$$\tau_{t,j} = C\ell_{t,j}/\|\boldsymbol{\ell}_t\|_2. \tag{19}$$

Once we achieve $\tau_{t,j}$, we can update every $f_{t+1,j}$ according to (2) and (8) at each round and achieve the fused classifier f_{t+1} on round t according to (9).

Note that there are mainly two ways of sharing knowledge among different sets of features. The first one is based on model parameters, such as coefficients in linear models, and the other one is based on features, which aims to learn common feature representations among different feature sets. Our algorithm belongs to the first type and we use a single global loss function to evaluate the quality of the multiple predictions and share knowledge about model parameters. Since it is challenging to learn common feature representations for different feature sets with unceasingly arriving instances, we have not employed the second type. Besides, the kernel functions used to handle nonlinear data would deteriorate the learning of common feature representations because we may have no explicit expression in the mapped high-dimensional space.

B. Budget Strategy

We can see from Section IV-A that when the mapping function $\phi(\cdot)$ of the used kernel does not have an explicit expression, the proposed method may need to store part of the past data, called SVs. As stated in [16], [42], [43], and [44], one critical issue in kernel-based online learning is the growing unbounded number of SVs with increasing computational and space complexity over time. Thus, to limit the amount of stored data when the mapping function $\phi(\cdot)$ of the used kernel does not have an explicit expression, we add a budget strategy, called random selection, which is commonly used in current research and enjoys the merits of simplicity and high efficiency [16], [42], [43].

Specifically, let $\mathcal{SV}_{t,j}$ denote the set of SVs for the jth classifier at round t and $|\mathcal{SV}_{t,j}|$ denote the set size. When the size of SVs exceeds the fixed budget B, i.e., $|\mathcal{SV}_{t,j}| > B$, we randomly select B SVs in $\mathcal{SV}_{t,j}$ uniformly to be stored and the rest are discarded. In summary, the procedure of our method is listed in Algorithm 1.

Algorithm 1 Proposed Algorithm IIF

```
1:  Input: parameter $C > 0$, budget $B \in N^+$;
2:  Initialization $\mathbf{w}_{1,1} = \cdots = \mathbf{w}_{1,T} = \mathbf{0}$, $|\mathcal{SV}_{1,1}| = \cdots = |\mathcal{SV}_{1,T}| = 0$;
3:  for $t = 1, 2, \ldots, T$ do
4:      Receive $\mathcal{X}_t$ and divide $\mathcal{X}_t$ into $t$ parts according to the way in Fig. 2, $\mathcal{X}_{t,j} \in \mathbb{R}^{n_t \times m_j}$, $j = 1, 2, \ldots, t$;
5:      Predict $\hat{\mathcal{Y}}_{t,j} = sign(f_{t,j}(\mathcal{X}_{t,j})) = (\hat{y}^1_{t,j}, \ldots, \hat{y}^{n_t}_{t,j})$;
6:      Receive correct labels $\mathcal{Y}_t = (y^1_t, \ldots, y^{n_t}_t)$;
7:      Calculate loss $\ell_{t,j}$ $(j = 1, 2, \ldots, t)$ according to Eq. (3);
8:      if $\exists\, \ell_{t,j} > 0$ then
9:          Calculate $\tau_{t,j}$ according to Eq. (19);
10:         Update $\mathcal{SV}_{t+\frac{1}{2},j} = \mathcal{SV}_{t,j} \cup (\mathcal{X}_{t,j}, \mathcal{Y}_t)$;
11:         if $|\mathcal{SV}_{t+\frac{1}{2},j}| > B$ then
12:             Randomly select $B$ support vectors uniformly from $\mathcal{SV}_{t+\frac{1}{2},j}$ and take them as $\mathcal{SV}_{t+1,j}$;
13:         else
14:             $\mathcal{SV}_{t+1,j} = \mathcal{SV}_{t+\frac{1}{2},j}$;
15:         end if
16:         Update $f_{t+1,j}$ according to Eq. (2) and Eq. (8);
17:     end if
18:     Update $f_{t+1}$ according to Eq. (9).
19: end for
```

V. Theoretical Analysis

We borrow the regret idea from online learning to derive a cumulative loss bound of the proposed algorithm. Two theorems and one lemma are given in this section. The two theorems discuss the upper bound of the cumulative hinge loss of the proposed algorithm in the case where $\tau_{t,j}$ adopts different definitions. The lemma is the crux to prove the above two theorems.

For clarity, we denote by $\ell_{t,j}$ the instantaneous loss of $\mathcal{X}_{t,j}$ suffered by $\mathbf{w}_{t,j}$ at round t and $\boldsymbol{\ell}_t$ the loss vector whose elements are $\ell_{t,j}$. Besides, we denote by $\tilde{\ell}_{t,j}$ the loss of $\mathcal{X}_{t,j}$ suffered by an off-line predictor $\tilde{\mathbf{w}}_j$ at round t and $\tilde{\boldsymbol{\ell}}_t$ the loss vector whose elements are $\tilde{\ell}_{t,j}$. To prove Theorems 1 and 2, we first prove a lemma that is the crux of our analysis.

Lemma 1: Let $\{(\mathcal{X}_{t,j}, \mathcal{Y}_t)\}^{1\leq j\leq t}_{1\leq t\leq T}$ be a sequence of data blocks, where $\mathcal{X}_{t,j} \in \mathbb{R}^{n_t \times m_j}$, and each $\mathbf{x}^i_{t,j} \subset \mathcal{X}_{t,j}$, $\mathbf{x}^i_{t,j} \in \mathbb{R}^{m_j}$, $y^i_{t,j} \in \{-1, +1\}$, and $\|\phi(\mathbf{x}^i_{t,j})\|_2 \leqslant R$. Let $\tilde{\mathbf{w}}_j \in \mathbb{R}^{m_j}$ be an arbitrary vector. The average hinge loss attained by $\tilde{\mathbf{w}}_j$ on $\mathcal{X}_{t,j}$ is defined to be $\tilde{\ell}_{t,j} = [(1/n_t)\sum_{i=1}^{n_t}(1 - y^i_{t,j}\tilde{\mathbf{w}}_j^T \cdot \phi(\mathbf{x}^i_{t,j}))]_+$. $\|\cdot\|$ is an absolute norm and $C > 0$. For the proposed algorithm IIF, we have

$$\sum_{t=1}^{T}\sum_{j=1}^{T}\left(2\tau_{t,j}\ell_{t,j} - \frac{\tau^2_{t,j}}{n_t^2}\left\|\sum_{i=1}^{n_t} y^i_{t,j}\cdot\phi\left(\mathbf{x}^i_{t,j}\right)\right\|_2^2\right) \leqslant \sum_{j=1}^{T}\left\|\tilde{\mathbf{w}}_j\right\|_2^2 + 2C\sum_{t=1}^{T}\left\|\tilde{\boldsymbol{\ell}}_t\right\|. \tag{20}$$

Proof: First, we define $\Delta_{t,j} = \|\mathbf{w}_{t,j} - \tilde{\mathbf{w}}_j\|_2^2 - \|\mathbf{w}_{t+1,j} - \tilde{\mathbf{w}}_j\|_2^2$. Then, we can prove the lemma by bounding $\sum_{t=1}^T\sum_{j=1}^T \Delta_{t,j}$. First, according to the definition of $\Delta_{t,j}$, for each $1 \leqslant j \leqslant T$, we can easily derive that

$$\sum_{t=1}^{T}\Delta_{t,j} = \left\|\mathbf{w}_{1,j} - \tilde{\mathbf{w}}_j\right\|_2^2 - \left\|\mathbf{w}_{T+1,j} - \tilde{\mathbf{w}}_j\right\|_2^2. \tag{21}$$

Obviously, $\|\mathbf{w}_{T+1,j} - \tilde{\mathbf{w}}_j\|_2^2 \geqslant 0$ and we can know from Algorithm 1 that $\mathbf{w}_{1,j} = (0, \ldots, 0)$ for all $1 \leqslant j \leqslant T$, so we have

$$\sum_{t=1}^{T}\sum_{j=1}^{T}\Delta_{t,j} \leqslant \sum_{j=1}^{T}\left\|\tilde{\mathbf{w}}_j\right\|_2^2. \tag{22}$$

$\sum_{j=1}^T \|\tilde{\mathbf{w}}_j\|_2^2$ is the upper bound of $\sum_{t=1}^T\sum_{j=1}^T \Delta_{t,j}$. Second, for its lower bound, we only consider $\Delta_{t,j} \neq 0$, which actually contributes to the sum. Since $\mathbf{w}_{t+1,j} = \mathbf{w}_{t,j} + (1/n_t)\tau_{t,j}\sum_{i=1}^{n_t} y^i_{t,j}\cdot\phi(\mathbf{x}^i_{t,j})$, we plug it into $\Delta_{t,j}$ and get

$$\begin{aligned}
\Delta_{t,j} &= \left\|\mathbf{w}_{t,j} - \tilde{\mathbf{w}}_j\right\|_2^2 - \left\|\mathbf{w}_{t+1,j} - \tilde{\mathbf{w}}_j\right\|_2^2\\
&= \left\|\mathbf{w}_{t,j} - \tilde{\mathbf{w}}_j\right\|_2^2 - \left\|\mathbf{w}_{t,j} + \frac{1}{n_t}\tau_{t,j}\sum_{i=1}^{n_t} y^i_{t,j}\cdot\phi\left(\mathbf{x}^i_{t,j}\right) - \tilde{\mathbf{w}}_j\right\|_2^2\\
&= \tau_{t,j}\left\{-\frac{2}{n_t}\mathbf{w}^T_{t,j}\sum_{i=1}^{n_t} y^i_{t,j}\cdot\phi\left(\mathbf{x}^i_{t,j}\right) - \frac{\tau_{t,j}}{n_t^2}\right.\\
&\quad\left.\times\left\|\sum_{i=1}^{n_t} y^i_{t,j}\cdot\phi\left(\mathbf{x}^i_{t,j}\right)\right\|_2^2 + \frac{2}{n_t}\tilde{\mathbf{w}}_j^T\sum_{i=1}^{n_t} y^i_{t,j}\cdot\phi\left(\mathbf{x}^i_{t,j}\right)\right\}\\
&= \tau_{t,j}\left\{\frac{2}{n_t}\sum_{i=1}^{n_t}\left(1 - y^i_{t,j}\cdot\mathbf{w}^T_{t,j}\cdot\phi\left(\mathbf{x}^i_{t,j}\right)\right)\right.\\
&\quad - \frac{\tau_{t,j}}{n_t^2}\left\|\sum_{i=1}^{n_t} y^i_{t,j}\cdot\phi\left(\mathbf{x}^i_{t,j}\right)\right\|_2^2\\
&\quad\left. - \frac{2}{n_t}\sum_{i=1}^{n_t}\left(1 - y^i_{t,j}\cdot\tilde{\mathbf{w}}_j^T\cdot\phi\left(\mathbf{x}^i_{t,j}\right)\right)\right\}.
\end{aligned}$$

As described above, we only consider nonzero summands, i.e., $\Delta_{t,j} \neq 0$. It implies that $\mathbf{w}_{t,j} \neq \mathbf{w}_{t+1,j}$ and $\ell_{t,j} \neq 0$. Thus, we can write $\ell_{t,j}$ as

$$\ell_{t,j} = \frac{1}{n_t}\sum_{i=1}^{n_t}\left(1 - y_{t,j}^i \cdot \mathbf{w}_{t,j}^T \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right).$$

Besides, according to the definition of $\tilde{\ell}_{t,j}$ in Lemma 1, it holds that

$$\tilde{\ell}_{t,j} \geqslant \frac{1}{n_t}\sum_{i=1}^{n_t}\left(1 - y_{t,j}^i \cdot \tilde{\mathbf{w}}_j^T \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right).$$

Then, we can easily derive that

$$-\tilde{\ell}_{t,j} \leqslant -\frac{1}{n_t}\sum_{i=1}^{n_t}\left(1 - y_{t,j}^i \cdot \tilde{\mathbf{w}}_j^T \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right).$$

Note that $\|\cdot\|$ is an absolute norm and $\ell_{t,j} \geqslant 0$, so $\tau_{t,j}$ defined in (19) is nonnegative, i.e., $\tau_{t,j} \geqslant 0$, we can derive the following inequality:

$$\begin{aligned}\Delta_{t,j} &= \tau_{t,j}\left\{\frac{2}{n_t}\sum_{i=1}^{n_t}\left(1 - y_{t,j}^i \cdot \mathbf{w}_{t,j}^T \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right)\right.\\ &\qquad - \frac{\tau_{t,j}}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2\\ &\qquad \left.- \frac{2}{n_t}\sum_{i=1}^{n_t}\left(1 - y_{t,j}^i \cdot \tilde{\mathbf{w}}_j^T \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right)\right\}\\ &= \tau_{t,j}\left\{2\ell_{t,j} - \frac{\tau_{t,j}}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2\right.\\ &\qquad \left.- \frac{2}{n_t}\sum_{i=1}^{n_t}\left(1 - y_{t,j}^i \cdot \tilde{\mathbf{w}}_j^T \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right)\right\}\\ &\geqslant \tau_{t,j}\left(2\ell_{t,j} - \frac{\tau_{t,j}}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2 - 2\tilde{\ell}_{t,j}\right).\end{aligned} \tag{23}$$

Obviously, summing (23) over $1 \leqslant j \leqslant T$, we can get

$$\sum_{j=1}^T \Delta_{t,j} \geqslant \sum_{j=1}^T\left(2\tau_{t,j}\ell_{t,j} - \frac{\tau_{t,j}^2}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2\right) - 2\sum_{j=1}^T \tau_{t,j}\tilde{\ell}_{t,j}. \tag{24}$$

In (24), $\sum_{j=1}^T \tau_{t,j}\tilde{\ell}_{t,j} = \boldsymbol{\tau}_t \cdot \tilde{\boldsymbol{\ell}}_t$. From (15), we know that for any $\boldsymbol{\tau}_t$, $\tilde{\boldsymbol{\ell}}_t$, it holds that $\boldsymbol{\tau}_t \cdot \tilde{\boldsymbol{\ell}}_t \leqslant \|\boldsymbol{\tau}_t\|^*\|\tilde{\boldsymbol{\ell}}_t\|$. Thus, we have $\sum_{j=1}^T \tau_{t,j}\tilde{\ell}_{t,j} \leqslant \|\tau_t\|^*\|\tilde{\boldsymbol{\ell}}_t\|$. Similarly, $\tau_{t,j}$ defined in (13) is clearly bounded by $\|\boldsymbol{\tau}_t\|^* \leqslant C$ and it is conservative, and we have that $\sum_{j=1}^T \tau_{t,j}\tilde{\ell}_{t,j} \leqslant C\|\tilde{\boldsymbol{\ell}}_t\|$. Combining this inequality with (24), we have

$$\sum_{j=1}^T \Delta_{t,j} \geqslant \sum_{j=1}^T\left(2\tau_{t,j}\ell_{t,j} - \frac{\tau_{t,j}^2}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2\right) - 2C\|\boldsymbol{\ell}_t\|^*. \tag{25}$$

Summing (25) over $1 \leqslant t \leqslant T$, we have

$$\sum_{t=1}^T\sum_{j=1}^T \Delta_{t,j} \geqslant \sum_{t=1}^T\sum_{j=1}^T\left(2\tau_{t,j}\ell_{t,j} - \frac{\tau_{t,j}^2}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2\right) - 2C\sum_{t=1}^T\|\boldsymbol{\ell}_t\|^*. \tag{26}$$

Combining (22) and (26), we are able to conclude the proof of Lemma 1. ■

With the help of Lemma 1, we are ready to prove the following two theorems, which discuss the upper bound of the cumulative hinge loss of the proposed algorithm in the case where $\tau_{t,j}$ adopts two different definitions.

Theorem 1: Let $\{(\mathcal{X}_{t,j}, \mathcal{Y}_t)\}_{1\leq t\leq T}^{1\leq j\leq t}$ be a sequence of data blocks, where $\mathcal{X}_{t,j} \in \mathbb{R}^{n_t \times m_j}$, and each $\mathbf{x}_{t,j}^i \subset \mathcal{X}_{t,j}$, $\mathbf{x}_{t,j}^i \in \mathbb{R}^{m_j}$, $y_{t,j}^i \in \{-1, +1\}$, and $\|\phi(\mathbf{x}_{t,j}^i)\|_2 \leqslant R$. Let $\tilde{\mathbf{w}}_j \in \mathbb{R}^{m_j}$ be an arbitrary vector. The average hinge loss attained by $\tilde{\mathbf{w}}_j$ on $\mathcal{X}_{t,j}$ is defined to be $\tilde{\ell}_{t,j} = [(1/n_t)\sum_{i=1}^{n_t}(1 - y_{t,j}^i\tilde{\mathbf{w}}_j{}^T \cdot \phi(\mathbf{x}_{t,j}^i))]_+$. If we define $\tau_{t,j}$ as the form of $\tau_{t,j} = C\ell_{t,j}/\|\boldsymbol{\ell}_t\|_2$, $C > 0$, then

$$\sum_{t=1}^T\|\boldsymbol{\ell}_t\| \leqslant \frac{1}{2C}\sum_{j=1}^T\left\|\tilde{\mathbf{w}}_j\right\|_2^2 + \sum_{t=1}^T\left\|\tilde{\boldsymbol{\ell}}_t\right\| + \frac{TR^2C\rho^2}{2} \tag{27}$$

where $\|\cdot\|$ is an absolute norm and ρ is the remoteness of $\|\cdot\|$.

Proof: According to Lemma 1, we can have

$$\sum_{t=1}^T\sum_{j=1}^T\left(2\tau_{t,j}\ell_{t,j} - \frac{\tau_{t,j}^2}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2\right) \leqslant \sum_{j=1}^T\left\|\tilde{\mathbf{w}}_j\right\|_2^2 + 2C\sum_{t=1}^T\left\|\tilde{\boldsymbol{\ell}}_t\right\|.$$

According to (18), we can rewrite the left-hand side of the above formula as

$$2C\sum_{t=1}^T\|\boldsymbol{\ell}_t\| - \sum_{t=1}^T\sum_{j=1}^T\frac{\tau_{t,j}^2}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2. \tag{28}$$

According to the assumption in Theorem 1 that $\|\phi(\mathbf{x}_{t,j}^i)\|_2^2 \leqslant R^2$, we have

$$\sum_{j=1}^T\frac{\tau_{t,j}^2}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2 \leqslant \sum_{j=1}^T\frac{\tau_{t,j}^2}{n_t^2}n_t^2R^2 = R^2\left\|\boldsymbol{\tau}_t\right\|_2^2. \tag{29}$$

From (16), we can know that $\|\boldsymbol{\tau}_t\|_2^2 \leq (\|\boldsymbol{\tau}_t\|^*)^2\rho^2$, so we can upper bound (29) by $R^2(\|\boldsymbol{\tau}_t\|^*)^2\rho^2$. Also, as stated before, $\tau_{t,j}$ has boundedness property, i.e., $\|\boldsymbol{\tau}_t\|^* \leqslant C$, so we are able to further bound (29) by $R^2C^2\rho^2$. Combining this bound with (28), we can have the following result:

$$2C\sum_{t=1}^T\|\boldsymbol{\ell}_t\| - \sum_{t=1}^T\sum_{j=1}^T\frac{\tau_{t,j}^2}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i \cdot \phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2 \geqslant 2C\sum_{t=1}^T\|\boldsymbol{\ell}_t\| - TR^2C^2\rho^2. \tag{30}$$

Combining (20) and (30), we can easily have

$$2C\sum_{t=1}^{T}\|\boldsymbol{\ell}_t\| - TR^2C^2\rho^2 \leqslant \sum_{j=1}^{T}\|\tilde{\mathbf{w}}_j\|_2^2 + 2C\sum_{t=1}^{T}\|\tilde{\boldsymbol{\ell}}_t\|. \quad (31)$$

Dividing both sides of the above formula by $2C$ and rearranging terms give the final bound, i.e.,

$$\sum_{t=1}^{T}\|\boldsymbol{\ell}_t\| \leqslant \frac{1}{2C}\sum_{j=1}^{T}\|\tilde{\mathbf{w}}_j\|_2^2 + \sum_{t=1}^{T}\|\boldsymbol{\ell}_t^*\| + \frac{TR^2C\rho^2}{2}. \quad (32)$$

Then, we conclude the proof of Theorem 1. ■

During the proof of Theorem 1, we have that the term

$$\sum_{j=1}^{T}\left(2\tau_{t,j}\ell_{t,j} - \frac{\tau_{t,j}^2}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i\cdot\phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2\right) \quad (33)$$

is lower bounded by $2C\|\boldsymbol{\ell}_t\| - R^2C^2\rho^2$. Obviously, $2C\|\boldsymbol{\ell}_t\|$ is proportional to $\|\boldsymbol{\ell}_t\|$ and $R^2C^2\rho^2$ is a constant. In practice, $2C\|\boldsymbol{\ell}_t\| - R^2C^2\rho^2$ may be negative when $\|\boldsymbol{\ell}_t\|$ is small enough, which implies that the value of the update step size $\tau_{t,j}$ may have been too large, and even more, it may imply that the update of classifier has increased the distance to the target.

Therefore, to derive an update for which (33) is always positive, we have added constraints to the domain of $\tau_{t,j}$. Specifically, we enforce $\tau_{t,j}$ to never be excessively large by limiting the dual norm of $\boldsymbol{\tau}_t$, which remains in the same direction as before. The definition of $\boldsymbol{\tau}_t$ is replaced by

$$\boldsymbol{\tau}_t = \underset{\boldsymbol{\tau}:\|\boldsymbol{\tau}\|^*\leqslant\min\{C,\|\boldsymbol{\ell}_t\|/R^2C^2\}}{\arg\max}\ \boldsymbol{\tau}\cdot\boldsymbol{\ell}_t. \quad (34)$$

Since $\|\cdot\|$ is a p-norm and $p=2$, we have

$$\tau_{t,j} = \begin{cases}\dfrac{\ell_{t,j}^{p-1}}{R^2\rho^2\|\boldsymbol{\ell}_t\|_p^{p-2}}, & \text{if } \|\boldsymbol{\ell}_t\|_p \le R^2C\rho^2\\[2ex] \dfrac{C\ell_{t,j}^{p-1}}{\|\boldsymbol{\ell}_t\|_p^{p-1}}, & \text{otherwise.}\end{cases} \quad (35)$$

In the following theorem, we prove the cumulative loss bound of IIF when $\tau_{t,j}$ adopts the definition in (35).

Theorem 2: Let $\{(\mathcal{X}_{t,j},\mathcal{Y}_t)\}_{1\le t\le T}^{1\le j\le t}$ be a sequence of data blocks, where $\mathcal{X}_{t,j}\in\mathbb{R}^{n_t\times m_j}$, and each $\mathbf{x}_{t,j}^i\subset\mathcal{X}_{t,j}$, $\mathbf{x}_{t,j}^i\in\mathbb{R}^{m_j}$, $y_{t,j}^i\in\{-1,+1\}$, and $\|\phi(\mathbf{x}_{t,j}^i)\|_2\leqslant R$. Let $\tilde{\mathbf{w}}_j\in\mathbb{R}^{m_j}$ be an arbitrary vector. The average hinge loss attained by $\tilde{\mathbf{w}}_j$ on $\mathcal{X}_{t,j}$ is defined to be $\tilde{\ell}_{t,j} = [(1/n_t)\sum_{i=1}^{n_t}(1-y_{t,j}^i\tilde{\mathbf{w}}_j{}^T\cdot\phi(\mathbf{x}_{t,j}^i))]_+$. If we define $\tau_{t,j}$ as the form of (35), $C>0$, then

$$1/\left(R^2\rho^2\right)\sum_{t\le T:\|\boldsymbol{\ell}_t\|\le R^2C\rho^2}\|\boldsymbol{\ell}_t\|^2 + C\sum_{t\le T:\|\boldsymbol{\ell}_t\|>R^2C\rho^2}\|\boldsymbol{\ell}_t\|^2 \le 2C\sum_{t=1}^{T}\|\tilde{\boldsymbol{\ell}}_t\| + \sum_{j=1}^{k}\|\tilde{\mathbf{w}}_j\|_2^2 \quad (36)$$

where $\|\cdot\|$ is an absolute norm and ρ is the remoteness of $\|\cdot\|$.

Proof: The first thing we need to figure out is whether $\tau_{t,j}$ defined in (35) has the properties of boundedness, nonnegativity, and conservativeness. Obviously, $\tau_{t,j}$ defined in (34) is bounded by $\|\boldsymbol{\tau}_t\|^*\leqslant C$ and it is conservative. Besides, since $\|\cdot\|^*$ is an absolute norm and $\ell_{t,j}\geqslant 0$, $\tau_{t,j}$ is nonnegative. Therefore, according to Lemma 1, we can have

$$\sum_{t=1}^{T}\sum_{j=1}^{T}\left(2\tau_{t,j}\ell_{t,j} - \frac{\tau_{t,j}^2}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i\cdot\phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2\right) \le \sum_{j=1}^{T}\|\tilde{\mathbf{w}}_j\|_2^2 + 2C\sum_{t=1}^{T}\|\tilde{\boldsymbol{\ell}}_t\|.$$

To prove Theorem 2, we need to lower bounding the left-hand side of the above formula. According to (35), we have to analyze two cases. First, if $\|\boldsymbol{\ell}_t\|\leqslant R^2C\rho^2$, then $\min\{C,\|\boldsymbol{\ell}_t\|/(R^2\rho^2)\} = \|\boldsymbol{\ell}_t\|/(R^2\rho^2)$. According to the definition of $\boldsymbol{\tau}_t$ and the fact that the dual of the dual norm is the original norm, we can have

$$2\sum_{j=1}^{T}\tau_{t,j}\ell_{t,j} = 2\|\boldsymbol{\tau}_t\|^*\|\boldsymbol{\ell}_t\| = 2\frac{\|\boldsymbol{\ell}_t\|^2}{R^2\rho^2}. \quad (37)$$

As proved in (29)

$$\sum_{j=1}^{T}\frac{\tau_{t,j}^2}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i\cdot\phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2$$

can be bounded by $R^2\|\boldsymbol{\tau}_t^2\|_2^2$ and $R^2\|\boldsymbol{\tau}_t^2\|_2^2$ can be bounded by $R^2(\|\boldsymbol{\tau}_t\|^*)^2\rho^2$ according to (16). Besides, according to the domain of $\|\boldsymbol{\tau}_t\|^*$, we have $\|\boldsymbol{\tau}_t\|^*\leqslant\|\boldsymbol{\ell}_t\|/(R^2\rho^2)$. Thus, we can bound $R^2(\|\boldsymbol{\tau}_t\|^*)^2\rho^2$ by $\|\boldsymbol{\ell}_t\|^2/(R^2\rho^2)$. In summary, we can have the following result:

$$\sum_{j=1}^{T}\frac{\tau_{t,j}^2}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i\cdot\phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2 \leqslant \frac{\|\boldsymbol{\ell}_t\|^2}{R^2\rho^2}. \quad (38)$$

From (20), (37), and (38), we can have the following result:

$$\frac{\|\boldsymbol{\ell}_t\|^2}{R^2\rho^2} \leqslant \sum_{j=1}^{T}\left(2\tau_{t,j}\ell_{t,j} - \frac{\tau_{t,j}^2}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i\cdot\phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2\right). \quad (39)$$

For the second case, if $\|\boldsymbol{\ell}_t\| > R^2C\rho^2$, then $\min\{C,\|\boldsymbol{\ell}_t\|/(R^2C^2)\} = C$. We have that

$$2\sum_{j=1}^{T}\tau_{t,j}\ell_{t,j} = 2\|\boldsymbol{\tau}_t\|^*\|\boldsymbol{\ell}_t\| = 2C\|\boldsymbol{\ell}_t\|. \quad (40)$$

As before, we can upper bound $R^2\|\boldsymbol{\tau}_t^2\|_2^2$ by $R^2(\|\boldsymbol{\tau}_t\|^*)^2\rho^2$. Using the fact that $\|\boldsymbol{\tau}_t\|^*\leqslant C$, we bound this term by $R^2C^2\rho^2$. Finally, using our assumption that $\|\boldsymbol{\ell}_t\| > R^2C\rho^2$, we conclude that

$$\sum_{j=1}^{T}\frac{\tau_{t,j}^2}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i\cdot\phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2 \leqslant C\|\boldsymbol{\ell}_t\|. \quad (41)$$

Subtracting both sides of this inequality from the respective sides of (40) gives

$$C\|\boldsymbol{\ell}_t\| \leqslant \sum_{j=1}^{T}\left(2\tau_{t,j}\ell_{t,j} - \frac{\tau_{t,j}^2}{n_t^2}\left\|\sum_{i=1}^{n_t} y_{t,j}^i\cdot\phi\left(\mathbf{x}_{t,j}^i\right)\right\|_2^2\right). \quad (42)$$

Comparing the upper bound in (20) with the lower bounds in (39) and (42) proves the theorem. ■

TABLE I
DETAILS OF DATASETS USED IN THE EXPERIMENTS

Dataset	# Inst.	# Feat.	Dataset	# Inst.	# Feat.
hearts	270	12	c-cancer	62	2000
p-gene	106	57	basehock	1993	4862
cleve	296	13	gisette	1000	5000
pima	768	8	PCMAC	1943	3289
spect	80	44	cifar	10000	3072
X8D5K1	400	8	rcv1	20242	47236
diabetes	768	8	RFID	940	150

VI. Experiments

In this section, we first introduce the datasets and the general settings. Then, we present the experimental results on both synthetic data and real-world applications.

A. Datasets and General Settings

We conduct our experiments on 12 datasets from UCI Repository[1] and LIBSVM Library,[2] and two real-world datasets that are rcv1 [45] and radio frequency identification (RFID) [46]. The details of the 14 datasets used in our experiments are listed in Table I.

Our experiments are conducted in four aspects. First, we compare IIF with seven state of the arts in Section VI-B. Second, in Section VI-C, we compare IIF with two traditional online learning methods, which can access all the features at each round for training. Third, we apply IIF on two real-world trapezoidal data streams in Section VI-D. Finally, we further evaluate the performance of IIF with respect to parameter C, an online variant of IIF, budget strategy, kernel function, and computational efficiency in Section VI-E–VI-I.

We simulate blocky trapezoidal data streams as follows. Each dataset is randomly divided into two nonoverlapping parts, with 50% as the training data and 50% as the testing data. Training data are used for blocky trapezoidal data stream learning. For the training data, we randomly split the data into T ($T = 2, 3, 4, 5, 6$) chunks, and each chunk corresponds to the instances received on each round. The current chunk not only contains all the features of the previous chunk but also randomly adds at least one feature. The feature dimension of the last chunk and the test data remains the same.

Unless otherwise specified, we use a linear kernel function in the proposed method, i.e., $\phi(\mathbf{x}_i) = \mathbf{x}_i$ and $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i{}^T\mathbf{x}_j$. After T rounds, we measure the performance of the learned classifier on test data. Both ACC and area under the curve (AUC) are used for evaluation metrics. For each dataset and each T, we run the experiment 20 times, each with a random partition. The results are reported by an average performance.

B. Comparisons With State of the Arts

In this section, we compare IIF with seven state of the arts. The first three methods are batch learning. Specifically, the first batch (FB) learns a classifier based on the data received on the first round, and then, it directly evaluates its performance on the test data. Similarly, the last batch (LB) learns a classifier based on the data received on the last round and random batch (RB) learns a classifier based on the data received on a randomly selected round. The first batch of features (FBF) uses the perceptron update strategy, it only uses the initial features to update its model, and the data stream in FBF is $(\mathcal{X}_{1,1}, \ldots, \mathcal{X}_{T,1})$. LibSVM is used to train a support vector machine (SVM) classifier for batch learning in FB, LB, and RB methods. The simple method is a simplified version of IIF, it does not consider the relationship between the classifiers, and the update of each classifier is carried out independently, without information exchange, i.e., $\mathbf{w}_{t+1,j} = \mathbf{w}_{t,j} + (1/n_t)\sum_{i=1}^{n_t} y_{t,j}^i \cdot \phi(\mathbf{x}_{t,j}^i)$. OL$_{\text{SF}}$ is a newly proposed trapezoidal data stream learning algorithm, with two variants OL$_{\text{SF}}$-I and OL$_{\text{SF}}$-II. According to [12], we use OL$_{\text{SF}}$-I for comparison and still mark it as OL$_{\text{SF}}$. OLI$_{\text{SF}}$-per is the compared method used in [12], which uses the perceptron update strategy. For methods FBF, OL$_{\text{SF}}$ and OLI$_{\text{SF}}$-per, we assume that the instance comes one by one, and they update the model on each instance. For OL$_{\text{SF}}$, the parameters are set according the description in [12]. Since all compared methods are not kernel-based methods, for fairness, we simply use the linear kernel in the proposed method, i.e., $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i{}^T\mathbf{x}_j$.

We present the empirical results of the above eight methods on 12 datasets. We set $T = 2, 3, 4, 5, 6$. The results of average ACC and AUC are shown in Table II and Fig. 3, respectively. It can be observed that the performance of IIF is better than or equal to the seven state of the arts on 12 datasets, which validates the effectiveness of our proposed methods. Specifically, for methods FB, LB, and RB, they perform poorly on most datasets, and it is because they only use one batch of the data streams to train a classifier, resulting in the loss of a large amount of data information. On a few datasets, they have good performance, such as FB on diabetes, LB on X8D5K1, and RB on gisette. This is because a good classifier can be trained without a large amount of training data on these datasets. Besides, the above three methods are quite unstable, which can be inferred from the large standard deviation of ACC values, such as gisette and hearts. Only selecting one batch of data streams for classifier learning cannot fully guarantee the stable performance of the classifier due to the randomness. The performances of methods FBF, simple, and OLI$_{\text{SF}}$-per are mixed. They work well on some datasets but perform poorly on others. FBF only uses the initial batch of features to update the model, still resulting in the loss of a large amount of data information. The only difference between simple and the proposed method IIF is whether the relationship between different classifiers is considered. As can be seen from Table II and Fig. 3, IIF is significantly better than simple, which shows the great importance of using a single global loss function to capture the relationship between different parts of data streams. Although OL$_{\text{SF}}$ performs better than the other six state of the arts on most datasets, our method has more wins than it. This is because in OL$_{\text{SF}}$, if the existing classifier correctly predicts the label of the current instance, the increased features will not be used for model update, resulting in a waste of feature information. Moreover, the standard deviation of the ACC values made by our method

[1]http://archive.ics.uci.edu/ml
[2]http://www.csie.ntu.edu.tw/~cjlin/libsvm

TABLE II

EXPERIMENTAL ACC [MEAN (STD)] RESULTS ON 12 DATASETS. THE BEST PERFORMANCE AND ITS COMPARABLE PERFORMANCES BASED ON PAIRED t-TESTS AT 95% SIGNIFICANCE LEVEL ARE HIGHLIGHTED IN BOLDFACE

Data sets	T	FB	LB	RB	FBF	Simple	$\mathrm{OLI_{SF}}$-per	$\mathrm{OL_{SF}}$	IIF
basehock	2	.495(.011)	.493(.010)	.495(.011)	.875(.022)	.714(.023)	.672(.036)	**.947(.015)**	**.947(.017)**
	3	.496(.010)	.496(.009)	.496(.009)	.853(.027)	.702(.031)	.651(.042)	**.945(.015)**	**.948(.014)**
	4	.520(.079)	.538(.114)	.564(.149)	.825(.031)	.673(.026)	.647(.039)	**.946(.013)**	**.952(.015)**
	5	.499(.009)	.520(.099)	.538(.115)	.835(.030)	.671(.031)	.661(.036)	**.947(.009)**	**.948(.014)**
	6	.507(.027)	.496(.009)	.496(.009)	.819(.024)	.666(.041)	.659(.031)	**.946(.008)**	**.949(.007)**
cifar	2	.499(.013)	.504(.012)	.501(.013)	**.712(.045)**	.680(.040)	.569(.056)	**.720(.013)**	**.728(.015)**
	3	.496(.014)	.499(.015)	.492(.012)	.648(.066)	.659(.070)	.583(.052)	**.709(.022)**	**.713(.022)**
	4	.499(.012)	.502(.012)	.495(.011)	.625(.053)	.633(.061)	.550(.059)	**.706(.016)**	**.713(.023)**
	5	.510(.023)	.524(.086)	.502(.010)	.617(.094)	.618(.083)	.596(.036)	**.714(.014)**	**.717(.019)**
	6	.494(.010)	.507(.009)	.501(.012)	.581(.083)	.558(.066)	.578(.060)	**.717(.011)**	**.724(.009)**
cleve	2	.696(.081)	**.785(.064)**	.723(.092)	.642(.059)	.727(.057)	.773(.035)	.774(.039)	**.801(.032)**
	3	.623(.093)	.748(.109)	.679(.115)	.624(.080)	.699(.045)	.764(.041)	.767(.033)	**.799(.025)**
	4	.580(.093)	.720(.133)	.662(.127)	.621(.098)	.698(.046)	.767(.039)	.749(.034)	**.798(.030)**
	5	.562(.023)	.678(.086)	.626(.010)	.562(.094)	.716(.083)	.759(.036)	.752(.014)	**.796(.019)**
	6	.560(.010)	.660(.009)	.536(.012)	.581(.083)	.680(.066)	.754(.060)	.752(.011)	**.794(.009)**
c-cancer	2	.613(.078)	.581(.101)	.590(.110)	.726(.103)	.690(.123)	.648(.083)	**.768(.075)**	**.784(.052)**
	3	.626(.088)	.545(.160)	.539(.093)	.723(.145)	.745(.136)	.661(.116)	**.797(.066)**	**.800(.071)**
	4	.497(.138)	.619(.088)	.568(.108)	.677(.087)	.690(.100)	.639(.128)	**.803(.064)**	**.810(.056)**
	5	.616(.144)	.523(.100)	.577(.125)	.713(.070)	.739(.103)	.710(.099)	**.777(.071)**	**.794(.071)**
	6	.697(.098)	.603(.145)	.677(.093)	.700(.113)	.700(.061)	.674(.084)	**.797(.040)**	**.816(.037)**
PCMAC	2	.490(.011)	.498(.015)	.492(.012)	.747(.063)	.654(.032)	.586(.024)	**.852(.015)**	**.860(.016)**
	3	.495(.013)	.501(.014)	.498(.014)	.737(.049)	.626(.024)	.580(.028)	**.852(.016)**	**.857(.011)**
	4	.506(.012)	.495(.013)	.523(.068)	.710(.044)	.631(.028)	.593(.023)	**.840(.017)**	**.847(.017)**
	5	.506(.012)	.493(.012)	.494(.012)	.713(.024)	.614(.018)	.594(.023)	**.838(.013)**	**.843(.023)**
	6	.504(.013)	.495(.013)	.494(.013)	.679(.043)	.601(.016)	.593(.023)	**.833(.008)**	**.843(.017)**
gisette	2	**.912(.014)**	**.915(.023)**	**.917(.022)**	.885(.026)	.499(.016)	.558(.059)	.695(.070)	**.901(.020)**
	3	**.875(.032)**	.865(.083)	**.882(.058)**	**.876(.032)**	.503(.014)	.518(.053)	.677(.078)	**.904(.029)**
	4	.841(.080)	**.878(.037)**	**.876(.031)**	**.888(.022)**	.501(.014)	.520(.046)	.681(.080)	**.900(.049)**
	5	.810(.074)	.857(.070)	.811(.090)	**.887(.021)**	.504(.013)	.524(.048)	.722(.069)	**.892(.063)**
	6	.705(.155)	.785(.095)	.776(.139)	.871(.031)	.498(.016)	.520(.036)	.681(.054)	**.912(.017)**
hearts	2	.691(.075)	**.820(.063)**	**.794(.077)**	.661(.076)	.733(.043)	.782(.045)	.789(.039)	**.827(.031)**
	3	.670(.077)	.776(.131)	.693(.125)	.630(.088)	.709(.022)	.779(.029)	.773(.041)	**.820(.030)**
	4	.619(.085)	.693(.131)	.689(.100)	.616(.117)	.746(.047)	.767(.050)	.783(.038)	**.814(.022)**
	5	.569(.093)	.701(.143)	.579(.112)	.579(.184)	.715(.060)	.769(.043)	.763(.035)	**.823(.028)**
	6	.609(.093)	.647(.122)	.584(.108)	.639(.107)	.729(.043)	.757(.046)	.752(.029)	**.809(.027)**
p-gene	2	.510(.073)	.480(.043)	.489(.057)	.624(.063)	.608(.101)	.675(.063)	**.680(.062)**	**.704(.048)**
	3	.532(.080)	.505(.064)	.515(.070)	.600(.072)	.606(.073)	.640(.061)	.639(.064)	**.694(.045)**
	4	.497(.059)	.497(.054)	.502(.060)	.602(.080)	.586(.068)	.657(.054)	.653(.069)	**.714(.046)**
	5	.492(.060)	.514(.060)	.493(.062)	.587(.058)	.612(.062)	**.687(.066)**	**.683(.076)**	**.704(.056)**
	6	.486(.042)	.496(.052)	.507(.068)	.532(.088)	.592(.075)	.631(.080)	.638(.074)	**.688(.070)**
diabetes	2	**.727(.025)**	**.736(.036)**	**.724(.035)**	.645(.075)	**.713(.019)**	.704(.026)	**.725(.020)**	**.724(.021)**
	3	**.720(.034)**	**.719(.051)**	**.720(.053)**	.644(.087)	**.720(.019)**	.701(.036)	**.722(.026)**	**.728(.014)**
	4	**.728(.032)**	.704(.047)	.706(.045)	.644(.127)	.723(.019)	.703(.035)	**.726(.019)**	**.732(.014)**
	5	**.721(.036)**	.655(.083)	.704(.046)	.631(.101)	**.716(.048)**	.700(.034)	**.726(.029)**	**.732(.018)**
	6	.712(.041)	.690(.043)	.689(.044)	.654(.092)	.724(.016)	.708(.026)	.728(.015)	**.734(.018)**
pima	2	.724(.029)	**.753(.020)**	.730(.033)	.653(.037)	.718(.019)	.710(.029)	.725(.021)	.724(.015)
	3	**.718(.034)**	**.724(.042)**	**.731(.028)**	.618(.105)	**.720(.018)**	.700(.031)	**.725(.022)**	**.727(.018)**
	4	**.727(.032)**	**.716(.042)**	**.723(.035)**	.624(.126)	**.718(.017)**	.698(.024)	**.722(.024)**	**.724(.023)**
	5	.712(.031)	.702(.044)	**.720(.031)**	.679(.088)	.715(.021)	.709(.029)	**.725(.023)**	**.729(.014)**
	6	.692(.042)	.680(.041)	.692(.048)	.643(.113)	.707(.024)	.700(.028)	**.719(.021)**	**.721(.017)**
spect	2	.464(.054)	.475(.090)	.473(.089)	.639(.084)	.689(.085)	**.747(.055)**	**.755(.056)**	**.777(.047)**
	3	.504(.067)	.506(.114)	.516(.118)	.576(.114)	.661(.078)	.720(.070)	**.734(.061)**	**.761(.055)**
	4	.512(.086)	.475(.086)	.475(.067)	.587(.105)	.686(.072)	**.717(.072)**	**.738(.073)**	**.749(.057)**
	5	.506(.071)	.541(.110)	.523(.076)	.569(.077)	.649(.097)	.701(.064)	.699(.077)	**.741(.079)**
	6	.503(.064)	.520(.100)	.507(.082)	.573(.080)	.701(.069)	.690(.074)	.703(.072)	**.734(.065)**
X8D5K1	2	**.999(.001)**	**1.00(.000)**	**1.00(.000)**	.998(.002)	**.998(.002)**	.999(.001)	**1.00(.000)**	**1.00(.000)**
	3	.998(.002)	**1.00(.000)**	**.999(.001)**	.996(.004)	.992(.007)	.999(.001)	.998(.002)	**1.00(.000)**
	4	.996(.004)	**1.00(.000)**	.999(.001)	.987(.013)	.996(.004)	.999(.001)	.999(.001)	**1.00(.000)**
	5	.991(.009)	**1.00(.000)**	**.999(.001)**	.995(.005)	.992(.008)	.998(.002)	.996(.004)	**1.00(.000)**
	6	.965(.033)	**1.00(.000)**	**.999(.001)**	.991(.008)	.992(.008)	.998(.002)	.998(.002)	**1.00(.000)**
win/tie/loss		51/9/0	46/13/1	47/13/0	56/4/0	54/6/0	57/3/0	26/34/0	–/–/–

in 20 repeats is also lower than all the state of the arts, which shows that our method has consistently better performance.

C. Comparisons With Traditional Online Learning Algorithms

In this section, we compare our method with two traditional online learning methods, i.e., budget passive–aggressive (BPA) online learning [26] and NOGD [27]. Specifically, we assume that BPA and NOGD can access all the features at each round for training, which indicates that the feature space is fixed. We try to explore how close the experimental results of our method are to the above two methods.

We can see from Fig. 4 that the differences between our method and the two traditional methods are relatively small. our method is even comparable to the above two methods on p-gene and spect datasets. It again proves that our method can effectively extract the information in the data streams and achieve satisfying results even if some features are missing.

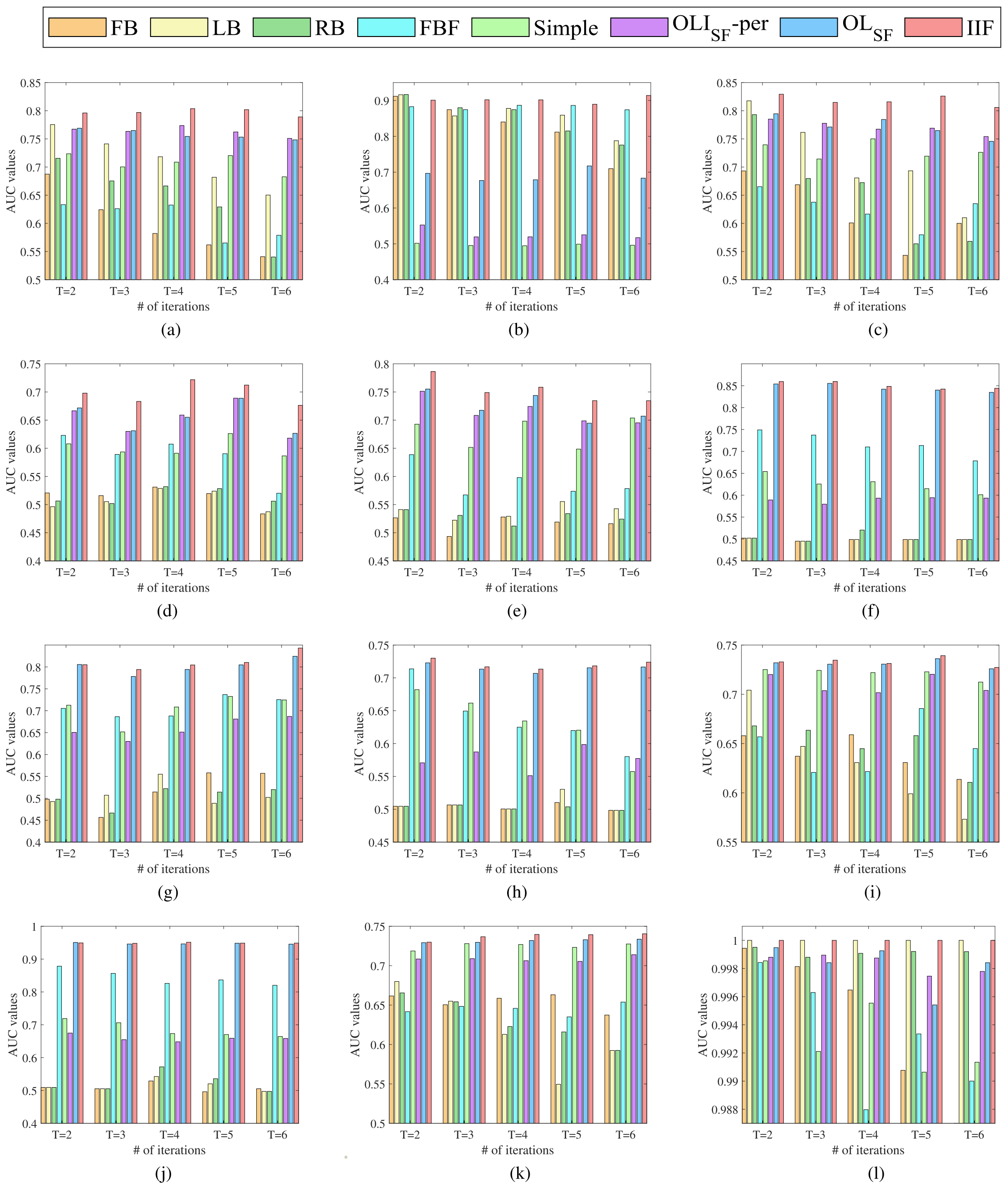

Fig. 3. AUC results for the 12 datasets. For each dataset, the bars from left to right represent the cases that $T = 2$, $T = 3$, $T = 4$, $T = 5$, and $T = 6$. (a) Cleve. (b) Gisette. (c) Hearts. (d) p-gene. (e) Spect. (f) PCMAC. (g) c-cancer. (h) Cifar. (i) Pima. (j) Basehock. (k) Diabetes. (l) X8D5K1.

D. Applications to Real-World Data Streams

In this section, we evaluate the performance of IIF on two real-world data streams, i.e., rcv1 [45] and RFID [46]. rcv1 aims to classify the JMLR articles into different groups. Since new articles are published continuously with new topics, this setting can be regarded as trapezoidal data stream learning. The "RFID" data stream is collected using the RFID technique. Each RFID aerial is used to receive the tag signals. To ensure continuous signal reception, new aerials are deployed next to the old ones before the old ones fail. During this overlapping period, data streams from both historical and augmented feature spaces can be achieved. Therefore, RFID also satisfies our assumptions.

According to the overall performance of all comparison methods in Section VI-B, we selected FBF, simple, OLI_{SF}-per, and OL_{SF} as representatives to compare with our method. Table III and Fig. 5 show the experimental results of the average ACC and AUC values. We can observe that IIF achieves significantly better results than the compared methods, and it not only performs better in classification but also has a more stable performance under different iterations, which can be verified from the smaller standard deviation.

E. Parameter Analysis

Here, we study the sensitivity of our algorithm to parameter C on four datasets, p-gene, basehock, PCMAC, and cifar.

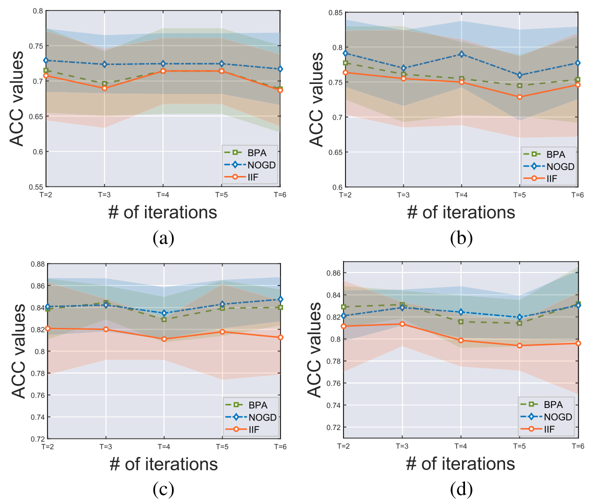

Fig. 4. ACC results on four datasets compared with two traditional online learning methods BPA and NOGD. (a) p-gene. (b) Spect. (c) Heart. (d) Cleve.

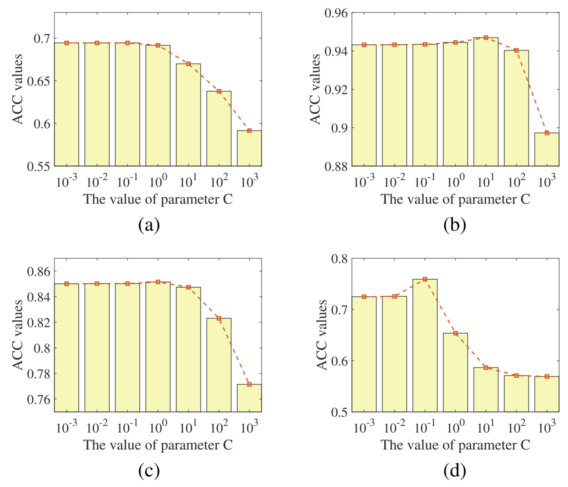

Fig. 6. ACC results on four datasets with respect to parameter C. (a) p-gene. (b) Basehock. (c) PCMAC. (d) Cifar.

TABLE III

EXPERIMENTAL ACC [MEAN (STD)] RESULTS ON TWO REAL-WORLD DATASETS COMPARED WITH FOUR STATE OF THE ARTS

Data sets	T	FBF	Simple	OLI$_{SF}$-per	OL$_{SF}$	IIF
RFID	2	.706(.057)	.731(.049)	.706(.040)	**.786(.033)**	**.806(.032)**
	3	.634(.037)	.686(.027)	.739(.037)	.767(.032)	**.791(.035)**
	4	.635(.060)	.679(.024)	.724(.037)	**.761(.040)**	**.784(.031)**
	5	.614(.056)	.661(.039)	.739(.037)	**.765(.030)**	**.783(.033)**
	6	.613(.056)	.665(.048)	.720(.049)	.759(.030)	**.787(.024)**
rcv1	2	.658(.020)	.683(.016)	.577(.071)	.692(.035)	**.863(.022)**
	3	.658(.021)	.653(.015)	.565(.075)	.712(.053)	**.865(.016)**
	4	.703(.047)	.652(.035)	.536(.102)	.646(.032)	**.856(.011)**
	5	.682(.036)	.646(.200)	.588(.089)	.686(.031)	**.859(.017)**
	6	.697(.046)	.638(.019)	.536(.102)	.648(.031)	**.856(.029)**
win/tie/loss		10/0/0	10/0/0	10/0/0	7/3/0	—/—/—

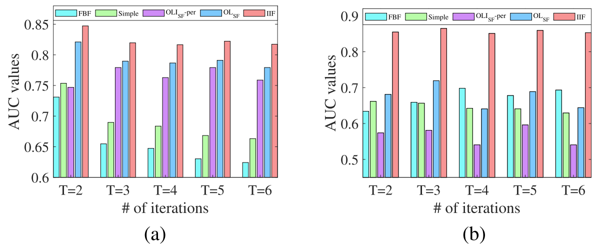

Fig. 5. ACC results on two real-world datasets compared with four state of the arts. (a) RFID. (b) rcv1.

As the common parameter setting methodology in online learning, we vary C in a wide range, i.e., $10^{[-3:1:3]}$, and set $T = 4$. As can be seen from Fig. 6, when the value of C is within a certain range, e.g., $0 < C < 0.1$, our method is not only insensitive to changes of C but also maintains good performance. The reason may be that C determines the updating step size, i.e., $\tau_{t,j} = C\ell_{t,j}/\|\boldsymbol{\ell}_t\|_2$. A small C can make the updating step size change smoothly and prevent turbulence in the optimization procedure. Certainly, a small C will delay the speed of convergence. Thus, to balance the computational efficiency and learning performance of the proposed method, we set $C = 0.01$ for all datasets in the experiments.

F. Comparison With Online Variant of IIF

In this section, we compare the proposed IIF with its online variant, named benchmark, to further demonstrate the effectiveness of blocky training. Specifically, when a data block $(\mathcal{X}_t, \mathcal{Y}_t)$ with n_t instances is observed, there are generally two ways to make use of them for model updating. One is our proposed blocky IIF, which uses all the instances in the block $(\mathcal{X}_t, \mathcal{Y}_t)$ to update the model simultaneously. The other is the online variant of IIF, named benchmark. It updates the model in an online manner, i.e., the model is updated immediately if we get one new instance in the data block.

TABLE IV

ACC [MEAN (STD)] RESULTS ON SIX DATASETS COMPARED WITH BENCHMARK

Data sets	Algorithms	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
basehock	Benchmark	.746(.031)	.844(.017)	.891(.018)	**.924(.014)**	**.940(.011)**
	IIF	**.776(.043)**	**.868(.017)**	**.907(.015)**	**.931(.012)**	**.942(.012)**
PCMAC	Benchmark	.659(.041)	.716(.028)	.767(.018)	.798(.015)	.823(.016)
	IIF	**.691(.050)**	**.740(.030)**	**.796(.017)**	**.816(.016)**	**.838(.015)**
gisette	Benchmark	**.821(.032)**	**.885(.025)**	**.910(.020)**	**.924(.022)**	**.935(.013)**
	IIF	**.825(.046)**	**.876(.034)**	.854(.101)	.909(.019)	.916(.015)
c-cancer	Benchmark	**.655(.132)**	**.706(.089)**	.700(.070)	**.710(.092)**	.787(.081)
	IIF	.632(.134)	**.690(.093)**	**.726(.078)**	**.723(.095)**	**.810(.075)**
RFID	Benchmark	**.654(.045)**	.667(.021)	**.703(.032)**	.750(.040)	.798(.024)
	IIF	**.656(.049)**	**.681(.023)**	**.710(.035)**	**.769(.043)**	**.814(.018)**
rcv1	Benchmark	.700(.031)	.773(.031)	.802(.018)	.814(.021)	.829(.021)
	IIF	**.725(.040)**	**.808(.021)**	**.827(.022)**	**.843(.019)**	**.857(.024)**

We conduct experiments on six datasets, i.e., basehock, PCMAC, gisette, c-cancer, RFID, and rcv1. We set $T = 5$. After each round, we measure the ACC performance of the learned classifiers on test data. We run the experiment 20 times, each with a random partition. Table IV shows the experimental results of the average ACC values. We can observe that the proposed algorithm IIF achieves better results than the benchmark on most datasets, which demonstrates that blocky training is better than online training.

G. Study on the Budget Strategy

In this section, we conduct experiments to study the role of budget strategy. As introduced in Section VI-B, we add a budget strategy to limit the amount of stored data when the mapping function $\phi(\cdot)$ of the used kernel does not have an explicit form, called random selection. Specifically, when the size of SVs exceeds a fixed budget B, we randomly select B SVs uniformly to be stored, and the rest will be discarded. We take the Gaussian kernel function widely used in the literature as an example and mark the budgeted IIF as

TABLE V

ACC [Mean (std)] Results of IIF_{RBF} and IIF_{RBF}-O Compared With Other State of the Arts When $B = 50$

Data sets	T	FBF	Simple	OLI_{SF}-per	OL_{SF}	IIF_{rbf}-O	IIF_{rbf}
RFID	2	.694(.059)	.718(.041)	.718(.039)	.770(.042)	**.879(.090)**	**.893(.074)**
	3	.645(.042)	.685(.035)	.716(.051)	.770(.028)	**.878(.057)**	**.888(.059)**
	4	.645(.040)	.678(.043)	.723(.037)	.769(.025)	**.867(.074)**	**.879(.070)**
	5	.605(.046)	.675(.040)	.748(.036)	.766(.026)	**.889(.076)**	**.893(.076)**
	6	.614(.034)	.667(.046)	.728(.036)	.756(.031)	**.900(.055)**	**.911(.047)**
cleve	2	.635(.089)	.698(.054)	.772(.043)	.777(.038)	**.796(.030)**	**.801(.020)**
	3	.620(.088)	.704(.037)	.757(.053)	.765(.049)	**.780(.051)**	**.792(.033)**
	4	.560(.126)	.689(.050)	.755(.041)	.761(.039)	**.768(.046)**	**.790(.035)**
	5	.617(.059)	.676(.059)	.765(.043)	**.776(.038)**	**.769(.054)**	**.790(.029)**
	6	.568(.086)	.698(.060)	.746(.028)	.769(.041)	.778(.032)	**.794(.021)**
hearts	2	.663(.049)	.731(.044)	.797(.035)	**.801(.041)**	**.806(.047)**	**.814(.031)**
	3	.643(.076)	.730(.041)	.773(.047)	.782(.041)	**.807(.048)**	**.813(.028)**
	4	.661(.096)	.702(.073)	.748(.043)	.762(.036)	**.795(.034)**	**.807(.026)**
	5	.621(.114)	.721(.033)	.761(.035)	.766(.033)	**.800(.063)**	**.810(.044)**
	6	.603(.161)	.736(.039)	.757(.049)	.759(.033)	.777(.048)	**.804(.025)**
p-gene	2	**.672(.064)**	.633(.052)	**.672(.074)**	**.677(.066)**	**.691(.069)**	**.687(.069)**
	3	.616(.061)	.588(.056)	**.647(.061)**	**.654(.066)**	**.664(.068)**	**.669(.063)**
	4	.561(.088)	.609(.066)	**.665(.059)**	**.673(.054)**	**.681(.076)**	**.684(.065)**
	5	.572(.081)	.604(.071)	**.654(.061)**	**.658(.060)**	**.673(.066)**	**.678(.063)**
	6	.551(.081)	.608(.079)	**.658(.074)**	**.660(.088)**	**.667(.094)**	**.682(.088)**
X8D5K1	2	.997(.003)	.999(.001)	**.999(.001)**	**1.00(.000)**	.998(.002)	**1.00(.000)**
	3	.995(.005)	.991(.009)	.999(.001)	.999(.001)	.993(.002)	**1.00(.000)**
	4	.982(.017)	.990(.010)	.998(.002)	**.999(.001)**	**1.00(.000)**	**1.00(.000)**
	5	.985(.014)	.994(.006)	.998(.002)	.998(.002)	**1.00(.000)**	**1.00(.000)**
	6	.990(.010)	.998(.002)	.998(.002)	**.999(.001)**	**1.00(.000)**	**1.00(.000)**
win/tie/loss		29/1/0	30/0/0	24/6/0	20/10/0	5/25/0	—/—/—

IIF_{rbf}. We randomly partition the data 20 times, and under each fixed data partition, we conduct multiple random selections by running our method 20 times, which is deliberately designed for the randomness of SVs' selection to obtain a reliable and trustful predictive performance. First, according to the overall performance of all comparison methods in Section VI-B, we select FBF, simple, OLI_{SF}-per, and OL_{SF} as representatives to compare with IIF_{rbf}. We conduct experiments on five datasets and set $B = 50$. The ACC results are shown in Table V. It can be observed that our method still outperforms other state of the arts, indicating its effectiveness with limited stored data.

Second, we study the sensitivity of our method to the budget B. We set $T = 4$. Fig. 7 shows the ACC results of our method under different values of parameter B on two datasets, i.e., RFID and cleve. We observe that when parameter B exceeds some thresholds, further increasement of B has a limited performance gain. This observation indicates that we can easily select a proper B to limit the amount of stored data while maintaining the superior performance of the method.

Finally, in addition to random selection, we employ another way, i.e., "Forgetron" [44], which discards the oldest SVs by assuming that an older SV is less representative of the distribution of fresh training data streams. It is incorporated into our method by replacing the random selection strategy and marked as IIF_{rbf}-O. We compare the random selection strategy IIF_{rbf} with IIF_{rbf}-O. Experimental results in Table V show that both IIF_{rbf} and IIF_{rbf}-O outperform the other state of the arts, and IIF_{rbf} has a better performance than IIF_{rbf}-O, which further indicates its effectiveness in alleviating the problem of data storage.

H. Comparison Between Linear Kernel and Gaussian Kernel

In this section, to explore the impact of different kernel functions on the performance of our method, we compare

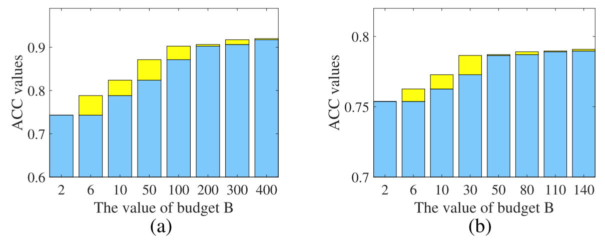

Fig. 7. ACC results of method IIF_{rbf} under different values of budget B. (a) RFID. (b) Cleve.

TABLE VI

ACC [Mean (std)] Results of IIF and IIF_{RBF} Algorithms on Four Datasets

Data sets	Algorithms	$T = 2$	$T = 3$	$T = 4$	$T = 5$	$T = 6$
cleve	IIF_{rbf}	**.791(.033)**	**.807(.038)**	**.793(.041)**	**.788(.027)**	**.786(.033)**
	IIF	**.792(.024)**	**.798(.032)**	**.790(.031)**	**.785(.030)**	**.788(.036)**
hearts	IIF_{rbf}	**.812(.025)**	**.826(.020)**	**.803(.028)**	**.803(.038)**	**.807(.031)**
	IIF	**.821(.016)**	**.827(.024)**	**.806(.026)**	**.808(.023)**	.802(.028)
spect	IIF_{rbf}	.710(.065)	.706(.058)	.695(.091)	.683(.088)	.665(.093)
	IIF	**.741(.055)**	**.746(.060)**	**.759(.077)**	**.764(.057)**	**.749(.063)**
RFID	IIF_{rbf}	**.884(.078)**	**.890(.056)**	**.897(.052)**	**.892(.047)**	**.899(.061)**
	IIF	.800(.043)	.798(.032)	.815(.021)	.814(.017)	.798(.023)

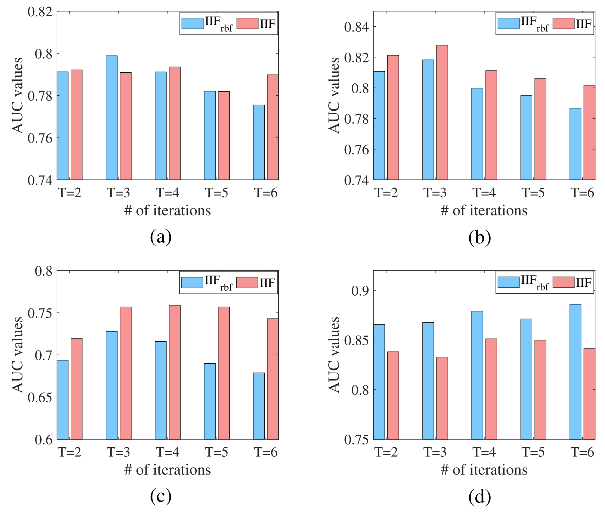

Fig. 8. AUC results of IIF and IIF_{rbf} on four datasets. (a) Cleve. (b) Hearts. (c) Spect. (d) RFID.

the performance of our method when using the Gaussian kernel (IIF_{rbf}) and linear kernel (IIF). We set $B = 50$ and randomly partition the data 20 times, and under each fixed data partition, we conduct multiple random selections by running IIF_{rbf} 20 times. The ACC and AUC results are shown in Table VI and Fig. 8, respectively. It can be observed that IIF_{rbf} does perform better than IIF on some datasets, but it is not consistently better on all datasets. The reason may be that the performance of different kernel functions is data-dependent. A more complicated kernel function does not always lead to better performance. Since the focus of this article is on the solution to this new setting of blocky trapezoidal data stream learning and the choice of the kernel function, or more commonly, model selection, is still a difficult problem in current research, we have not specified kernel selection in this article. In the future, we will conduct in-depth research in this special setting.

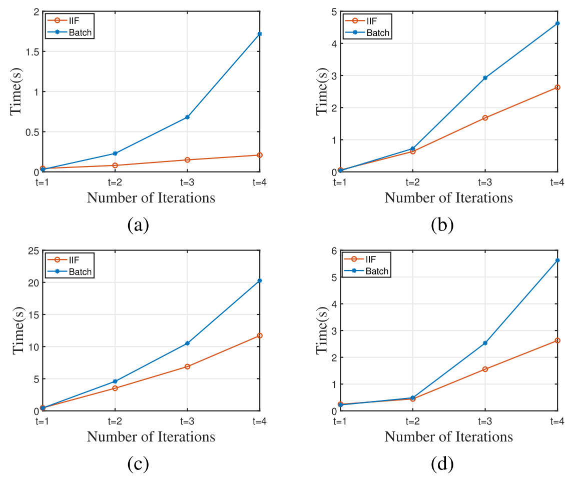

Fig. 9. Curve shows the CPU time for IIF and batch IIF algorithms over the number of iterations. (a) Gisette. (b) PCMAC. (c) Cifar. (d) Basehock.

TABLE VII
AVERAGE CPU TIME OF THE COMPARED METHODS
(O-per REPRESENTS OLI_{SF}-per)

Data	Algorithms							
	FB	LB	RB	FBF	Simple	O-per	OL_{SF}	IIF
cleve	0.0004	0.0004	0.0004	0.0003	0.0016	0.0024	0.0027	0.0019
c-cancer	0.0004	0.0006	0.0005	0.0002	0.0018	0.0043	0.0045	0.0022
PCMAC	0.0971	0.2650	0.1450	0.0047	1.4600	0.1620	0.1700	1.4800
gisette	0.0341	0.1490	0.0981	0.0044	0.7500	0.1440	0.1410	0.7580
hearts	0.0003	0.0003	0.0003	0.0002	0.0014	0.0022	0.0025	0.0015
basehock	0.0414	0.4750	0.4970	0.0033	2.3900	0.2500	0.2530	2.4100
cifar	3.8700	8.5100	6.1700	1.0426	11.120	2.4990	2.4990	12.100
p-gene	0.0002	0.0002	0.0002	0.0002	0.0008	0.0011	0.0011	0.0008
diabetes	0.0005	0.0006	0.0005	0.0002	0.0034	0.0053	0.0060	0.0035
pima	0.0006	0.0006	0.0005	0.0003	0.0041	0.0058	0.0067	0.0044
spect	0.0002	0.0002	0.0002	0.0001	0.0007	0.0009	0.0009	0.0007
X8D5K1	0.0003	0.0002	0.0002	0.0002	0.0017	0.0021	0.0033	0.0018

I. Comparisons on Computational Efficiency

In this section, we demonstrate the computational efficiency of our method through the following two aspects. First, according to the strategy used in [47] and [48], we conduct comparison experiments of CPU time for incremental learning algorithm IIF and batch IIF. Specifically, when a new data block is obtained at iteration $t, t = 1, \ldots, T$ (here, we set $T = 4$), for batch IIF (abbreviated as batch), the construction of the classification model is achieved from scratch, it will repeat the learning from the beginning and the knowledge acquired in the past is discarded, while for IIF, the classification model is updated from the previously learned model and the newly obtained data block. Thus, the CPU time for IIF to construct the classification model is just the updating time. The experimental results are presented in Fig. 9. We can observe that as t increases, the CPU time gap between IIF and batch IIF keeps increasing, which demonstrates that the proposed incremental learning algorithm IIF has the advantage of fast and timely updating over the batch learning method.

Second, we compare the average CPU time for IIF and the seven state of the arts on 12 datasets. Table VII shows the average time elapsed for data stream learning. Similarly, we set $T = 4$. As can be seen from Table VII, our method is slower than OLI_{SF}-per and OL_{SF} on relatively large datasets but has an obvious speed advantage than these two methods on most datasets. Although our method takes more runtime than most compared methods, it achieves a good balance between classification accuracy and operating speed, which makes our method more effective for practical applications.

VII. CONCLUSION

In this article, we aim to learn a highly dynamic classification model from blocky trapezoidal data stream and propose a new algorithm called IIF. This method first gradually increases the independently updated classifiers based on the data steam obtained on each round. Then, the predictive quality of each individual classifier can be evaluated by the single global loss function. The final classifier is achieved by using the idea of ensemble. For the applicability of this method, we directly transform it into the kernel method in this article. We theoretically prove the worst case relative loss bounds for IIF. Empirical studies on various synthetic data and real-world applications validate the effectiveness of our algorithm.

In fact, the assumption about blocky trapezoidal data streams does not always hold in reality due to the detectors' limited lifespan, which will make the features that correspond to the old detectors vanish. Therefore, a more realistic assumption is that the historical features vanish arbitrarily or that each arriving data stream can arbitrarily carry features. Besides, it is also necessary to explore the relationship between features of different parts from other aspects, such as learning common feature representations. These interesting assumptions are quite useful in practice and we will do further study on them.

REFERENCES

[1] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. 20th Int. Conf. Mach. Learn.*, Washington, DC, USA, 2003, pp. 928–936.
[2] T. Roughgarden and O. Schrijvers, "Online prediction with selfish experts," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, vol. 30, 2017, pp. 1300–1310.
[3] J. Xu, Y. Y. Tang, B. Zou, Z. Xu, L. Li, and Y. Lu, "The generalization ability of online SVM classification based on Markov sampling," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 3, pp. 628–639, Mar. 2015.
[4] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 551–585, Dec. 2006.
[5] J.-W. Liu, J.-J. Zhou, M. S. Kamel, and X.-L. Luo, "Online learning algorithm based on adaptive control theory," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2278–2293, Jun. 2018.
[6] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.
[7] P. Zhou, P. Li, S. Zhao, and X. Wu, "Feature interaction for streaming feature selection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4691–4702, Oct. 2020.
[8] S. Perkins and J. Theiler, "Online feature selection using grafting," in *Proc. 20th Int. Conf. Mach. Learn.*, Washington, DC, USA, 2003, pp. 592–599.
[9] X. Wu, K. Yu, W. Ding, H. Wang, and X. Zhu, "Online feature selection with streaming features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 5, pp. 1178–1192, May 2013.
[10] J. Wang et al., "Online feature selection with group structure analysis," 2016, *arXiv:1608.05889*.
[11] C. Hou, L.-L. Zeng, and D. Hu, "Safe classification with augmented features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 9, pp. 2176–2192, Sep. 2019.
[12] Q. Zhang, P. Zhang, G. Long, W. Ding, C. Zhang, and X. Wu, "Online learning from trapezoidal data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 10, pp. 2709–2723, Oct. 2016.
[13] E. Beyazit, J. Alagurajah, and X. Wu, "Online learning from data streams with varying feature spaces," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 3232–3239.

[14] D. Wu, Y. He, X. Luo, M. Shang, and X. Wu, "Online feature selection with capricious streaming features: A general framework," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Los Angeles, CA, USA, Dec. 2019, pp. 683–688.
[15] O. Dekel, P. M. Long, and Y. Singer, "Online multitask learning," in *Proc. 19th Annu. Conf. Learn. Theory (COLT)*. Pittsburgh, PA, USA: Springer, vol. 4005, 2006, pp. 453–467.
[16] S. C. H. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *Neurocomputing*, vol. 459, pp. 249–289, Oct. 2021.
[17] J. Wang, P. Zhao, and S. C. H. Hoi, "Soft confidence-weighted learning," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 1, p. 15, 2016.
[18] A. Cutkosky and K. A. Boahen, "Online convex optimization with unconstrained domains and losses," 2017, *arXiv:1703.02622*.
[19] S. Kale, Z. S. Karnin, T. Liang, and D. Pál, "Adaptive feature selection: Computationally efficient online sparse linear regression under RIP," in *Proc. 34th Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, vol. 70, 2017, pp. 1780–1788.
[20] K. Jun, A. Bhargava, R. D. Nowak, and R. Willett, "Scalable generalized linear bandits: Online computation and hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, vol. 30, 2017, pp. 99–109.
[21] V. N. Vapnik, "Statistical learning theory," *Encyclopedia Ences Learn.*, vol. 41, no. 4, p. 3185, 1998.
[22] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Mach. Learn.*, vol. 37, no. 3, pp. 277–296, 1999.
[23] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.
[24] Z. Wang, K. Crammer, and S. Vucetic, "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale SVM training," *J. Mach. Learn. Res.*, vol. 13, pp. 3103–3131, Oct. 2012.
[25] J. Lu, P. Zhao, and S. C. H. Hoi, "Online sparse passive aggressive learning with kernels," in *Proc. SIAM Int. Conf. Data Mining*, Miami, FL, USA, 2016, pp. 675–683.
[26] Z. Wang and S. Vucetic, "Online passive-aggressive algorithms on a budget," in *Proc. 30th Int. Conf. Artif. Intell. Statist.* Sardinia, Italy: Chia Laguna Resort, vol. 9, 2010, pp. 908–915.
[27] J. Lu, S. C. H. Hoi, J. Wang, P. Zhao, and Z. Liu, "Large scale online kernel learning," *J. Mach. Learn. Res.*, vol. 17, p. 47, Apr. 2016.
[28] T. D. Nguyen, T. Le, H. Bui, and D. Phung, "Large-scale online kernel learning with random feature reparameterization," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Melbourne, VIC, Australia, Aug. 2017, pp. 2543–2549.
[29] L. Jian, S. Shen, J. Li, X. Liang, and L. Li, "Budget online learning algorithm for least squares SVM," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 9, pp. 2076–2087, Sep. 2017.
[30] K. Crammer, J. S. Kandola, and Y. Singer, "Online classification on a budget," in *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8–13, 2003, Vancouver and Whistler, British Columbia, Canada]*, S. Thrun, L. K. Saul, and B. Schölkopf, Eds. Cambridge, MA, USA: MIT Press, 2003, pp. 225–232. [Online]. Available: https://proceedings.neurips.cc/paper/2003/hash/1a68e5f4ade56ed1d4bf273e55510750-Abstract.html
[31] J. Lu, D. Sahoo, P. Zhao, and S. C. H. Hoi, "Sparse passive-aggressive learning for bounded online kernel methods," *ACM Trans. Intell. Syst. Technol.*, vol. 9, no. 4, pp. 1–27, Jul. 2018.
[32] T. Luo, C. Hou, F. Nie, and D. Yi, "Dimension reduction for non-Gaussian data by adaptive discriminative analysis," *IEEE Trans. Cybern.*, vol. 49, no. 3, pp. 933–946, Mar. 2019.
[33] C. Hou and Z.-H. Zhou, "One-pass learning with incremental and decremental features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 11, pp. 2776–2792, Nov. 2018.
[34] B. Hou, Y. Yan, P. Zhao, and Z. Zhou, "Storage fit learning with feature evolvable streams," 2020, *arXiv:2007.11280*.
[35] Z. Zhang, P. Zhao, Y. Jiang, and Z. Zhou, "Learning with feature and distribution evolvable streams," in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, vol. 119, 2020, pp. 11317–11327.
[36] B.-J. Hou, L. Zhang, and Z.-H. Zhou, "Prediction with unpredictable feature evolution," 2019, *1904.12171*.
[37] Y. He, B. Wu, D. Wu, E. Beyazit, and X. Wu, "Toward mining capricious data streams: A generative approach," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 3, pp. 1228–1240, Mar. 2020.
[38] J. A. Sáez, M. Galar, J. Luengo, and F. Herrera, "Analyzing the presence of noise in multi-class problems: Alleviating its influence with the one-vs-one decomposition," *Knowl. Inf. Syst.*, vol. 38, no. 1, pp. 179–206, 2014.
[39] J. Xu, "An extended one-versus-rest support vector machine for multi-label classification," *Neurocomputing*, vol. 74, no. 17, pp. 3114–3124, Oct. 2011.
[40] C. M. Bishop and N. M. Nasrabadi, "Pattern recognition and machine learning," *J. Electron. Imag.*, vol. 16, no. 4, 2007, Art. no. 049901.
[41] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2012.
[42] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile, "Tracking the best hyperplane with a simple budget perceptron," *Mach. Learn.*, vol. 69, no. 2, pp. 143–167, 2007.
[43] S. C. H. Hoi, J. Wang, P. Zhao, R. Jin, and P. Wu, "Fast bounded online gradient descent algorithms for scalable kernel-based online learning," in *Proc. 29th Int. Conf. Mach. Learn. (ICML)*, Edinburgh, U.K., 2012.
[44] O. Dekel, S. Shalev-Shwartz, and Y. Singer, "The forgetron: A kernel-based perceptron on a fixed budget," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2008, pp. 1–8.
[45] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, p. 27, May 2011, [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm
[46] B. Hou, L. Zhang, and Z. Zhou, "Learning with feature evolvable streams," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, vol. 30, 2017, pp. 1417–1427.
[47] X. Zhang, L. Cheng, D. Chu, L.-Z. Liao, M. K. Ng, and R. C. E. Tan, "Incremental regularized least squares for dimensionality reduction of large-scale data," *SIAM J. Sci. Comput.*, vol. 38, no. 3, pp. B414–B439, Jan. 2016.
[48] D. Chu, L.-Z. Liao, M. K. P. Ng, and X. Wang, "Incremental linear discriminant analysis: A fast algorithm and comparisons," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 11, pp. 2716–2735, Nov. 2015.

Shilin Gu received the B.S. and M.S. degrees from the National University of Defense Technology, Changsha, China, in 2017 and 2019, respectively, where he is currently pursuing the Ph.D. degree.

His research interests include machine learning and data mining.

Yuhua Qian (Member, IEEE) received the M.S. and Ph.D. degrees in computers with applications from Shanxi University, Taiyuan, China, in 2005 and 2011, respectively.

He is currently a Professor with the Key Laboratory of Computational Intelligence and Chinese Information Processing, Ministry of Education, Shanxi University. He is best known for multigranulation rough sets in learning from categorical data and granular computing. He is involved in research on machine learning, pattern recognition, feature selection, granular computing, and artificial intelligence. He has authored over 100 articles on these topics in international journals.

Dr. Qian served on the Editorial Board of the *International Journal of Knowledge-Based Organizations* and *Artificial Intelligence Research*. He has served as the Program Chair or Special Issue Chair for the Conference on Rough Sets and Knowledge Technology, the Joint Rough Set Symposium, and the Conference on Industrial Instrumentation and Control, and a PC member for many machine learning and data mining conferences.

Chenping Hou (Member, IEEE) received the Ph.D. degree from the National University of Defense Technology, Changsha, China, in 2009.

He is currently a Full Professor with the Department of Systems Science, National University of Defense Technology. He has authored more than 80 peer-reviewed papers in journals and conferences, such as the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TSMCB/TCB, IEEE TRANSACTIONS ON IMAGE PROCESSING, IJCAI, and AAAI. His current research interests include machine learning, data mining, and computer vision.